ON THE NONLINEAR COMMUTED SYNTHESIS OF THE PIANO

Balázs Bank and László Sujbert

Department of Measurement and Information Systems
Budapest University of Technology and Economics, Hungary
bank@mit.bme.hu, sujbert@mit.bme.hu

ABSTRACT

In this paper a novel method is presented for the physics-based sound synthesis of the piano, based on digital waveguides. The approach combines the advantages of the commuted synthesis technique and the methods using a nonlinear hammer model. The interaction force of the hammer-string contact is computed by an auxiliary digital waveguide connected to a nonlinear hammer model. This force signal is used as a target impulse response for designing a low-order digital filter real-time. The piano sound is calculated by filtering the soundboard response with the hammer filter and feeding this signal to a synthesizer digital waveguide. A new method is presented for separating the contribution of the interaction force and the soundboard in measured piano tones. For modeling beating, a new technique is proposed based on a simplified pitch-shift effect. Considerations on modeling the effect of sustain pedal are also given. It is shown that the technique of designing the hammer filter real-time is not only useful for digital waveguide modeling, but it can be combined with sampling synthesis too.

1. INTRODUCTION

The earliest piano model applying digital waveguides was presented in 1987 [1], using a semi-physical hammer model. Fully physical interaction models for the piano were presented later [2, 3]. The advantage of these approaches is that the dynamic properties of the hammer-string interaction are easily reproduced. Thus, the model reacts to the change of impact velocity in a physically meaningful way. A high-order system is needed for simulating the radiation effect of the soundboard properly. The piano models based on the straightforward technique provide good sonic results, except the attack of the notes. This is a serious shortcoming, since the attack is a distinctive property of piano sound.

A piano model using commuted synthesis was presented in [4, 5], where the hammer is modeled as a linear filter. Therefore, the components of the system can be commuted. The soundboard filter is implemented as a wavetable, whose content is fed to the digital waveguide through a hammer filter. The wavetable of soundboard response is computed by soundboard measurements with impact hammer excitation or modeled by white noise led trough a time-variant filter. This way, the same problem arises with the attack of the tones as for the straightforward technique. An advantage is that no soundboard filter is needed. As a drawback, different hammer filters must be designed off-line for all the notes and hammer parameters, which makes the interaction of the musician hard to take into account.

Here a novel synthesis method is presented, combining the advantages of the previous approaches. For the synthesis, the approach of commuted piano is taken, but the hammer filters are designed real-time. The target response for the filter design is computed by an auxiliary digital waveguide connected to a nonlinear hammer model. This way, a fully physical approach is used for that part of the model only which is controlled by the musician. The soundboard wavetable is computed from recorded piano tones by inverse filtering, resulting in proper reproduction of the attack.

The paper is organized as follows: first, the basic ideas of string and hammer modeling are presented. This is followed by the description of the proposed nonlinear commuted piano model. Three strategies for designing and implementing the hammer filter are described in a separate section. After that, some developments are presented to the basic model concerning string coupling. Connections to sampling synthesis are also outlined. Summary and future plans conclude the paper.

2. STRING AND HAMMER MODELING

2.1. The digital waveguide

The most efficient approach for string instrument modeling is the digital waveguide [6]. The method is based on the time-domain solution of the one-dimensional wave equation. By discretizing the traveling wave solution with respect to time and space, the model reduces to a pair of delay lines. By assuming linearity, losses and dispersion of the string can be lumped to one termination and taken into account by digital filters. When the loss and dispersion filters and the two delay lines are consolidated, the model becomes a filter and a delay line in a feedback loop. The transfer function of such a model is the following:

$$S_s(z) = \frac{1}{1 - z^{-N} H(z)} \tag{1}$$

where $H(z) = H_l(z)H_d(z)H_{fd}(z)$ is the loop filter made up of three parts. The filter $H_l(z)$ is responsible for modeling the losses and $H_d(z)$ for modeling the inharmonicity of the string. The fractional delay filter $H_{fd}(z)$ is used for fine-tuning the fundamental frequency of the digital waveguide model. This structure $S_s(z)$ will be used in this study for synthesizing the tone with the commuted synthesis approach.

When the behavior of a specific point on the string has to be computed, this model has to be amended by simple comb filters. This is the case when a hammer model is connected to the digital waveguide. The interaction force $F(n)$, which forms the basis of hammer filter design, will be computed by the auxiliary waveguide $S_a(z)$ in this way.

2.2. The hammer model

Inspite of its simplicity, considering the hammer as a mass connected to a nonlinear spring seems to describe the behavior of real hammers amazingly well [7]. The continuous-time equations are the following:

$$F(t) = f(\Delta x(t)) = \begin{cases} K[\Delta x(t)]^p & \text{if } \Delta x(t) > 0 \\ 0 & \text{if } \Delta x(t) \leq 0 \end{cases} \quad (2)$$

$$F(t) = -m_h \frac{d^2 x_h(t)}{dt^2} \quad (3)$$

where $F(t)$ is the interaction force, $\Delta x(t) = x_h(t) - x_s(t)$ is the compression of the hammer felt, where $x_h(t)$ and $x_s(t)$ are the positions of the hammer and the string, respectively. The hammer mass is referred by m_h, K is the hammer stiffness coefficient, and p is the stiffness exponent ranging from 2 to 4. Due to this nonlinearity, the tone spectrum varies dynamically with hammer velocity.

Eqs. (2) and (3) have to be discretized with respect to time to fit for connecting the hammer to the digital waveguide model. However, there is a mutual interdependence between two variables: the hammer force $F(t)$ depends on the hammer displacement $x_h(t)$ according to Eq. (2), while $x_h(t)$ is computed from $F(t)$ by Eq. (3). This results in a delay-free loop, which cannot be implemented directly. Inserting an additional delay element in the loop may results in numerical instability. This can be solved by rearranging the equations with the K-method [2] or by increasing the sampling rate of the hammer model [3]. In this study, the multi-rate approach of [3] is used.

3. THE NONLINEAR COMMUTED PIANO

3.1. Model structure

The model structure is presented in Fig. 1. First, the auxiliary waveguide $S_a(z)$ connected to a nonlinear hammer model is run to compute the interaction force $F(n)$. This string model $S_a(z)$ can be relatively simple, i.e., the dispersion and loss filters can be omitted, since they do not influence the interaction force $F(n)$ significantly. The string model $S_a(z)$ can be even simplified to one single delay line, modeling the reflected pulses coming from the agraffe, since the reflected pulses from the far end of the string usually return only after the hammer has left the string. After the hammer leaves the string (typically after 100 samples at $f_s = 44.1$ kHz), the operation of the auxiliary model is stopped, requiring a small amount of computation only, meaning about 500 operations. Note that in this study "number of operations" refers to the approximate number of MAC (multiply and accumulate) instructions, since we are considering DSP implementation.

Next, a hammer filter $H_h(z)$ is designed real-time. As the simplest solution, $F(n)$ can be directly implemented as an FIR filter. More efficient approaches will be presented in Sec. 4. The hammer filter $H_h(z)$ is then used for commuted synthesis: the soundboard response $y_{sb}(n)$ is filtered through $H_h(z)$ and lead to the synthesizer waveguide model $S_s(z)$. This has to be done until the end of the wavetable is reached: afterwards the string model $S_s(z)$ vibrates freely. This means that both the reading from the wavetable and the filtering with $H_h(z)$ is stopped. Note that the synthesizer waveguide can start to sound only after $F(n)$ is computed and the filter $H_h(z)$ is designed, leading to a small amount of latency in the system.

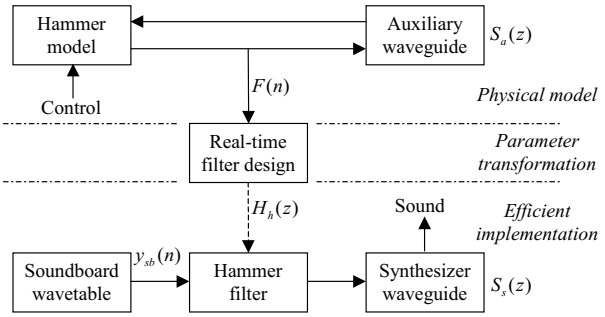

Figure 1: *Model structure of the nonlinear commuted piano.*

3.2. Parameter estimation

In this section, we describe the parameter estimation done off-line. The hammer filter design, which is done during the synthesis real-time, will be outlined in Sec. 4.

Originally, the motivation of using commuted synthesis was eliminating the need of soundboard filters. Here the reason is rather having a proper attack for the synthesized piano tones. Therefore, the approach of inverse filtering [8] is used. First, the loop filter of the synthesizer waveguide $S_s(z)$ is designed from recorded piano tones by the methods described in [9] and [10]. The recorded sound $y(n)$ is filtered by the inverse of the synthesizer waveguide $1/S_s(z)$. This can be done, since the inverse filter $1/S_s(z)$ is always stable, when the corresponding loop filter $H(z)$ is stable, as can be seen from Eq. (1). Now the inverse filtered signal $y_i(n)$ combines the contributions of the soundboard and the hammer-string interaction.

The effects of the soundboard and the hammer strike have to be separated, since they will correspond to different parts of the piano model of Fig. 1. We propose the following: first, the interaction force $y_h(n)$ is approximately measured by attaching a small accelerometer to the piano hammer [7], parallel with recording the sound pressure signal $y(n)$. Then, $y_h(n)$ is used to deconvolve the inverse filtered signal $y_i(n)$ to produce the soundboard contribution $y_{sb}(n)$, since by our definition $y_i(n) = y_{sb}(n) * y_h(n)$, where $*$ denotes convolution.

Now the signal $y_{sb}(n)$ after windowing (typically up to 2000 samples) can be used as the content of the wavetable, since the contribution of the hammer and the string are filtered out. Note that this procedure could be also used for improving the quality of (linear) commuted piano models. The benefit of the inverse-filtering approach is that if the impulse response of the hammer filter $H_h(z)$ is the same as the measured force $y_h(n)$, the synthesized tone becomes exactly like the original up to the length of the soundboard wavetable, preserving the attack. The drawback is that separate wavetables are needed for each note of the piano.

Unfortunately, the string model $S_s(z)$ is only a rough approximation of the real string behavior. Thus, $y_{sb}(n)$ will contain harmonic components besides the response of the soundboard. Due to this, when the end of the wavetable is reached, the sound changes significantly. If the wavetable is short and if its last part is faded out smoothly, this change is inaudible. Nevertheless, it becomes a problem when we try to improve the quality of the attack transients by increasing the length of the soundboard wavetable (e.g., up to 5000 samples). This can be avoided by separating the har-

monic (or deterministic) part $y_{det}(n)$ of the recorded piano sound from the residual transient part $y_{res}(n)$ prior to inverse filtering [8]. Now the inverse filtered harmonic and residual parts can be windowed to different lengths. Thus, the length of the excitation table can be increased without having any effect on the harmonic contents of the tone.

4. THE HAMMER FILTER

As the hammer filter $H_h(z)$ should be designed real-time, not only the filter should be simple, but also the design algorithm, to create only a small latency in the system. Many approaches can be taken, we present here three examples working well in practice. These three techniques lead to different sound quality and different computational requirements.

4.1. FIR filter

As a straightforward solution, the hammer force $F(n)$ computed by the auxiliary model can be directly implemented as an FIR filter. The "filter design" takes no computation at all, and the hammer filter $H_h(z)$ is perfectly accurate, but computationally heavy. Depending on the length of $F(n)$, filtering consumes about 100 operations per sample. This filtering has to be done in the beginning of the tones, that is, until the end of soundboard wavetable is reached, which is typically after 2000 samples, leading to a total number of filtering operations of 200000. This way, the average load of the signal processing device remains still acceptable. However, it might be a problem that the computational load is uneven: each note requires about 5 times more computation at its first 2000 samples compared to the samples computed after the hammer filtering is ended.

As an example, a DSP running at 40 MHz allows about 1000 instructions per sample. If the half its power is used for running the synthesizer string models, 500 instructions remain for hammer filtering. This means that 5 hammer filters can be computed in the same time, i.e., five tones can be started within 2000 samples (ca. 50 ms), meaning 100 tones in a second, which is appropriate for most of the musical pieces. However, when more than 5 notes are started in the same time by playing a chord, some notes have to wait as much as 50 ms to start. Obviously, this can be completely avoided if we allow the simultaneous computation of 10 hammer filters, since the pianist can press maximum 10 keys in the same time, but this increases the computational requirement even more.

4.2. IIR filter

The attack sound (the knock) of the high notes can be perfectly rendered only by using longer wavetables ($\approx$ 5000 samples), and in this case the FIR filtering approach of Sec. 4.1 is too demanding. This is the case for those applications as well, where 1000 instructions per sample are not available. Therefore, we present here a different solution based on IIR filters, strongly reducing the computational demand.

The hammer filter $H_h(z)$ is now made up of two parts: the first is an integrating FIR filter $H_{h,i}(z)$ providing an impulse response of linear segments [11]. This is connected in series with a second filter $H_{h,e}(z)$, which is a low-order IIR filter equalizing the response. The integrating FIR filter $H_{h,i}(z)$ is designed to produce an output connecting the local minima and maxima of $F(n)$ with straight lines. The filter $H_{h,i}(z)$ consists of a sparsely tapped delay line and two integrators connected in series. In this case, the delay line is implemented by having multiple reading pointers in the wavetable. The effect of the two integrators is precomputed in the soundboard wavetable.

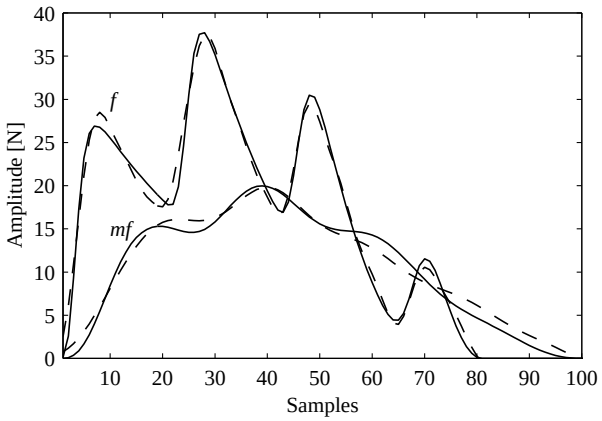

Figure 2: *Hammer filter design: the target response $F(n)$ (solid line) and the impulse response of the hammer filter $H_h(z)$ (dashed line) for playing forte and mezzoforte (the latter is multiplied by a factor of 5 for picture clarity) on a C_4 string.*

The output of $H_{h,i}(z)$ is smoothed by the equalization filter $H_{h,e}(z)$, which is a low order IIR filter computed by least-squares filter design. The equalization filter $H_{h,e}(z)$ is calculated by minimizing the L_2 norm of the error vector $\mathbf{e}$:

$$\mathbf{e} = \mathbf{M}\mathbf{p} - \mathbf{y} \qquad \mathbf{p}_{opt} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{y} \tag{4}$$

where $\mathbf{y}$ is equal to the desired output $F(n)$ and $\mathbf{p}$ is the parameter vector containing the coefficients of $H_{h,e}(z)$. The matrix $\mathbf{M}$ consists of the past values of the desired output $F(n)$ and the present and past values of input, which is the hammer pulse computed by linear segments with $H_{h,i}(z)$. The solution $\mathbf{p}_{opt}$ is obtained by simple least-squares minimization. Note that this is optimal in the least-squares sense for FIR filters only. For designing IIR filters, the previous output values in $\mathbf{M}$ are not the real outputs of the designed filter but the outputs given by the specification. This distorts our results, but practice shows that the designed IIR filters are well behaving. More sophisticated system identification approaches would lead to more accurate filters, but higher computational demand on filter design.

As for computational requirements, the most demanding parts of Eq. (4) are $\mathbf{M}^T\mathbf{M}$ and $\mathbf{M}^T\mathbf{y}$, taking $MN^2 + MN$ operations, where M is the length of $\mathbf{y}$ and N is the length of $\mathbf{p}$. As an example, the response of the hammer filter $H_h(z) = H_{h,i}(z)H_{h,e}(z)$ is depicted in Fig. 2, dashed line for a C_4 piano note at two dynamic levels. Here a 4^{th} order all-pole filter $H_{h,e}(z)$ is used for equalization, requiring about 3000 instructions for filter design. Computing the parameters of $H_{h,i}(z)$ takes about 300 instructions. Running the filter $H_h(z)$ for synthesis consumes 15-20 instructions, depending on the number of linear segments in $H_{h,i}(z)$.

The latency due to calculating $F(n)$ and designing $H_h(z)$ strongly depends on how many notes are started in the same time. Let us assume that 100 operations per sample are available for hammer filter design, which is 10 % of the computational power of a DSP running at 40 MHz. The required ca. 4000 operations

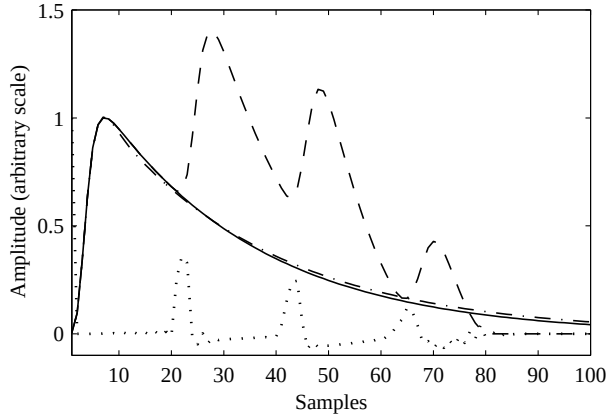

Figure 3: *Designing hammer filter based on decomposing the force signal $F(n)$ (dashed line) to the single pulse of an infinite string $F_s(n)$ (solid line) and the contribution of the reflections $F_r(n)$ (dotted line). The dash-dotted line refers to the response of the 3rd order IIR filter designed in the least-squares sense by Eq. (4). For better visibility, the signals are scaled proportionately such that the maximal value of $F_s(n)$ has been set to unity.*

result in a 40 sample delay, which is less then 1 ms at $f_s = 44.1$ kHz for one note. If 10 notes are started together, this means 10 ms, which is still inaudible. Increasing the computational power used for hammer filter design reduces this delay even more.

4.3. IIR filter based on force signal decomposition

In some special cases, the computational requirement of 15-20 operations per sample for hammer filtering can be still high. This is the case for modeling the effect of the sustain pedal, where the filtering with $H_h(z)$ has to be done for the length of the whole note (see Sec. 5.2.2). Therefore, we present an even simpler solution, based on the idea proposed in [5].

When the hammer hits an infinite string, where no reflected pulses are present, the hammer force consists in one single pulse $F_s(n)$ with a sharp attack and a slow, nearly exponential decay, as it can be seen in Fig. 3, solid line. If the string is terminated, the reflected pulses will interfere with the original, producing multiple maxima in the force signal. This is depicted by dashed line in Fig. 3. By deconvolving the force signal $F(n)$ with the impulse of the unterminated string $F_s(n)$, the contribution of the reflected pulses $F_r(n)$ can be obtained (dotted line in Fig. 3), since we define $F_r(n)$ such that $F(n) = F_r(n) * F_s(n)$. In [5] this idea was used for designing the hammer filters off-line, and the two different parts were implemented as separate filters. The filter corresponding to $F_r(n)$ was kept the same for all the different dynamic levels of a note, and only the behavior of the single pulse $F_s(n)$ was varied depending on hammer velocity.

We have found that $F_r(n)$ varies significantly by dynamic level, so it should be kept constant only when sound quality requirements are lower. If we hold $F_r(n)$ constant, the soundboard wavetable can be prefiltered by its response. During synthesis, only the single pulse $F_s(n)$ is computed and the hammer filter is now designed from this pulse real-time. The least-squares approach of Eq. 4 is used for designing the hammer filter $H_h(z)$ similarly to Sec. 4.2, but now the input of the system identification is a unit pulse and the output is the single pulse of $F_s(n)$. As it can be seen in Fig. 3, a 3rd order IIR filter produces almost an exact match of the response. The real-time least-squares filter design consumes about 5600 operations, and the 3rd order IIR filter takes 7 filtering instructions per sample for each note.

The latency, estimated as described at the end of Sec. 4.2 is about 60 samples, i.e., 1.5 ms at $f_s = 44.1$ kHz for one note. If 10 notes are started together, it is raised to 15 ms at a computational load of 100 operation per sample.

4.4. Comparison of filter design approaches

Table 1 shows the estimated computational costs of the filtering approaches presented in this section. The example is the C_4 note taken in Figs. 2 and 3, the length of the force signal $F(n)$ is 100 samples long. The table shows that the total number of computations for one note are greatly reduced by taking the IIR filtering approach of Sec. 4.2 compared to FIR filtering. This difference is even higher when the length of wavetable is 5000 samples. The sound quality decreases slightly compared to using FIR filters. The IIR filter design of Sec. 4.3 based on decomposition reduces the computational requirements even more, but the sound quality drops as well. Therefore, we suggest using the method of Sec. 4.2 for synthesis and the method of Sec. 4.3 for special effects, such as modeling the sustain pedal in Sec. 5.2.2.

	FIR	IIR	Decomp. IIR
Filter design	0	3000	5600
Filtering (ops./sample)	100	20	7
Total (WT ln. = 2000)	200000	43000	19600
Total (WT ln. = 5000)	500000	103000	40600

Table 1: *Estimated number of operations for the three different hammer filter approaches of Sec. 4.1-4.3. The "WT ln." refers to the length of the soundboard wavetable in samples.*

5. FURTHER DEVELOPMENTS

As all the strings of the piano are attached to the same bridge-soundboard system, their vibrations are inherently coupled. This coupling can be considered at two levels: one is the coupling between strings belonging to one key, resulting in beating and two-stage decay. The other is the coupling between different notes, producing the sustain-pedal effect and determining the characteristic sound of the undamped high strings. By modeling these phenomena the sound quality of the piano model can be improved significantly. This will be outlined in the next sections.

5.1. Beating and two-stage decay

Beating and two-stage decay is a distinctive characteristic of piano sound, coming from the interaction of the two transversal polarizations and the two or three strings belonging to one single key. This can be simulated by running more string models in parallel, with the appropriate coupling [12]. Although the method provides accurate results when implemented in the frequency domain, designing coupling filters for time domain modeling is still an unsolved

problem. Therefore, we describe here two simpler approaches, considering the problem rather from the perceptual point of view.

5.1.1. The multi-rate resonator bank

Informal listening tests show that the existence of beating in piano sound changes the perceived quality significantly, but the evolution of partial envelopes do not have to be precisely modeled. Based on this fact, the multi-rate resonator bank was presented in [13]. There a resonator bank is run parallel with the basic string model $S_s(z)$. It consists of 5 to 10 resonators, whose frequency, decay time, initial amplitude and phase parameters can be set separately. Accordingly, only those partial envelopes are simulated precisely, where the beating and the two-stage decay are significant, the others have simple exponential decay determined by the basic string model $S_s(z)$. By running the resonators at a lower sampling rate, the computational demand is reduced to 5–10 instructions per sample.

The parameter estimation is done off-line by analyzing the partial envelopes of the recorded tones. It is relatively simple, since there is no need for coupling filter design, and the stability problem of the coupled strings is also avoided. A drawback that the parameters of the resonator bank are not physically meaningful, thus they cannot be changed intuitively by the user.

5.1.2. The pitch-shift method

Here we present a new method for simulating beating in a very simple way. The idea is based on the fact that when two independent string models are sounded together mistuned by 1 or 2 cents (where 100 cents correspond to one semitone), it already produces quite natural beating sound. Obviously, the same effect can be obtained, when the signal of one string model is mixed to its pitch-shifted version. Pitch-shifting could be applied as a post-processing technique for all the notes together, but having different mistuning for the different notes, i.e., having separate pitch-shifters for all the notes produces more natural sound.

The simplest pitch-shifter is a circular buffer, whose content is read slower than it is written. Ideally, it is a lengthening delay line. The problem with pitch shifters is that when the delay line gets too long, the reading pointer has to jump back by a certain amount, producing some artifacts. In this special case this is simple, since the period length of the tone is known, so the pointer can jump back exactly one period. If more memory is available, this "jump back" can be even omitted: the lengthening delay line gets only 1000 samples long for 2 cents of mistuning after 20 seconds at $f_s = 44.1$ kHz. This simplifies the structure even more.

In practice, the delay line has to be completed by a fractional delay filter. We have found that linear interpolation is suitable for that purpose. Accordingly, modeling beating can be implemented in a very simple way. The transfer function of the structure is the following:

$$H_{ps} = 1 + c\left((1-d)z^{-D} + dz^{-(D+1)}\right) \qquad (5)$$

where $D + d$ is the total lenght of the delay made up of an integer part D and a fractional part $0 \leq d < 1$. The mixing parameter $0 \leq c \leq 1$ controls the intensity of beating. From Eq. (5) comes that the technique can also be considered as a time-varying comb filter, i.e., a special type of flanger effect.

An advantage of the technique is that the user has two intuitive parameters: the amount of mistuning and the ratio of mixing the two signals. Clearly, the beating is not modeled precisely, but the sound quality is significantly improved when compared to the model without beating simulation.

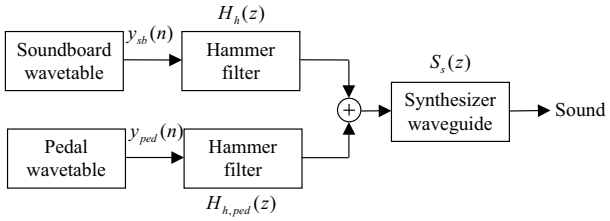

Figure 4: *Model structure for synthesizing the sustain-pedal effect. Note that blocks required for hammer filter design are not depicted.*

5.2. The sustain-pedal effect

Another important feature of the piano sound is the sustain-pedal effect. When this pedal is depressed, the dampers of all the strings are opened, thus all the strings are excited by the sounding notes. Here we describe two strategies for modeling this phenomenon.

5.2.1. Coupling of all the strings

In [2], all the string models were coupled to the same termination, which was modeled by a complex filter having multiple resonances. This produces natural pedal effects when all the 88 string models of the piano are run parallel, but this might be too demanding for some applications.

5.2.2. The commuted pedal effect

In [5] the sustain-pedal effect was proposed to be simulated by recording the soundboard response with open strings and using that as the soundboard wavetable. Here we suggest the combination of the original, inverse filtered wavetable $y_{sb}(n)$ and a common sustain-pedal signal $y_{ped}(n)$.

The soundboard response with the dampers lifted is recorded by a force hammer. This may be done at two or three different positions, since the response varies significantly when the low and high notes are compared. Then, the contribution of the soundboard has to be separated from that of the coupled string vibrations, since it is already included in $y_{sb}(n)$. This is done by measuring the response at the same position with the dampers down and by subtracting the dry response from the one measured with open strings.

At the synthesis, the contents of the soundboard wavetable $y_{sb}(z)$ and the pedal effect $y_{ped}(n)$ are added together before led to the string model $S_s(z)$. However, $y_{ped}(n)$ may be 20 seconds long, therefore filtering with a complicated hammer filter of Sec. 4.2 is too demanding from a computational point of view. Therefore, we suggest to compute the hammer pulse $F_s(n)$ and design the filter real-time as described in Sec. 4.3 and use that as hammer filter $H_{h,ped}(z)$ for $y_{ped}(n)$. This is displayed in Fig. 4. The contribution of the reflected pulses $F_r(n)$ now cannot be used to prefilter $y_{ped(z)}$, since $F_r(n)$ is different for all the notes and now we are using only one common pedal wavetable $y_{ped}(n)$ for one region of the soundboard. The general shape of the spectrum is determined by $F_s(n)$ anyway, and the sustain pedal is a secondary

effect, therefore omitting $F_r(n)$ does not influence sound quality significantly.

The commuted pedal-effect is computationally much less demanding compared to the coupled strings approach presented in [2]. However, it has one serious drawback: if a note is already started, the pedal effect cannot be added afterwards. In other words, only those notes can have proper sustain-pedal effect, which were started after the sustain pedal was depressed. Some tricks might be used, e.g., starting to read the contents of pedal wavetable $y_{ped}(n)$ from the position corresponding to the time elapsed after the note has been started to sound, but this is far from an exact solution.

Nevertheless, the commuted pedal-effect can be used for modeling the behavior of the highest octave without any drawbacks. There no dampers are present, therefore all the high strings start to sound with open strings around. The approach remains the same as outlined above, but now the soundboard response around the region of high strings has to be measured with the dampers down. The contribution of the soundboard can be separated from that of the sympathetic string vibrations by a second measurement with all the high strings damped by rubber or felt.

6. CONNECTIONS TO SAMPLING

In sampling synthesis, the separate tones of the piano are recorded and then played back by looping the sustained part after the attack. Time-variant filters and amplifiers account for the spectral change as a function of time. For the variation of the timbre with respect to key velocity, simple lowpass filters are used. These usually do not produce the feel of real piano dynamics.

The method presented here could be used for improving the quality of sampling synthesizers by replacing the dynamics filter with $H_h(z)$ calculated by a real-time physical model. In this case, the wavetables of the sampler would contain piano samples inverse filtered with the impact force $y_h(n)$, measured as outlined in Sec. 3.2. Besides leading to better dynamic behavior, the system would provide the user a more intuitive control on the timbre by adjusting hammer parameters instead of filter coefficients.

7. CONCLUSION

The paper presented the idea of nonlinear commuted piano by combining the advantages of previous approaches. The method provides a sound quality comparable to sampling synthesizers at an affordable computational cost, while providing a "physical" control for the musician. Three different techniques were presented for the design and implementation of the hammer filters. Different techniques were described for modeling the string coupling and the pitch-shift based beating model was proposed as the most efficient implementation. The sustain-pedal effect was also considered and the technique was found to be useful in modeling the sympathetic vibrations of the open strings in the high register.

As for future work, different approaches could be developed and compared for hammer filter design. Modeling the sustain pedal with the commuted approach for notes started before the dampers have been raised is also an open problem. Considerations have to be made how the restrike of the same string should be modeled. The proper modeling of the effect of dampers is also a part of future plans.

8. ACKNOWLEDGEMENTS

This work was started during Balázs Bank was a research assistant at Padua University, Italy. The visit was funded by the EU project MOSART Improving Human Potential. The work at Budapest University of Technology and Economics was supported by the Hungarian National Scientific Research Fund OTKA under contract number F035060.

9. REFERENCES

[1] Guy E. Garnett, "Modeling piano sound using digital waveguide filtering techniques," in *Proc. Int. Computer Music Conf.*, Urbana, Illinois, USA, 1987, pp. 89–95.

[2] Gianpaolo Borin, Davide Rocchesso, and Francesco Scalcon, "A physical piano model for music performance," in *Proc. Int. Computer Music Conf.*, Thessaloniki, Greece, September 1997, pp. 350–353.

[3] Balázs Bank, "Physics-based sound synthesis of the piano," M.S. thesis, Budapest University of Techology and Economics, Hungary, May 2000, Published as Report 54 of HUT Laboratory of Acoustics and Audio Signal Processing, URL: `http://www.mit.bme.hu/~bank`.

[4] Julius O. Smith and Scott A. Van Duyne, "Commuted piano synthesis," in *Proc. Int. Computer Music Conf.*, Banff, Canada, September 1995, pp. 335–342.

[5] Scott A. Van Duyne and Julius O. Smith, "Developments for the commuted piano," in *Proc. Int. Computer Music Conf.*, Banff, Canada, September 1995, pp. 319–326.

[6] Julius O. Smith, "Physical modeling using digital waveguides," *Computer Music J.*, vol. 16, no. 4, pp. 74–91, Winter 1992.

[7] Xavier Boutillon, "Model for piano hammers: Experimental determination and digital simulation," *J. Acoust. Soc. Am.*, vol. 83, no. 2, pp. 746–754, February 1988.

[8] Vesa Välimäki and Tero Tolonen, "Development and calibration of a guitar synthesizer," *J. Aud. Eng. Soc.*, vol. 46, no. 9, pp. 766–778, September 1998.

[9] Davide Rocchesso and Francesco Scalcon, "Accurate dispersion simulation for piano strings," in *Proc. Nordic Acoust. Meeting*, Helsinki, Finland, 1996, pp. 407–414.

[10] Balázs Bank and Vesa Välimäki, "Robust loss filter design for digital waveguide synthesis of string tones," *to be published in the IEEE Signal Processing Letters*.

[11] Robert L. M. Heylen and Malcolm O. Hawksford, "The integrating finite impulse response filter," in *Proc. 94th AES conv., Preprint No. 3587*, Berlin, Germany, March 1993.

[12] Mitsuko Aramaki, Julien Bensa, Laurent Daudet, Philippe Guillemain, and Richard Kronland-Martinet, "Resynthesis of coupled piano string vibrations based on physical modeling," *Journal of New Music Research*, vol. 30, no. 3, pp. 213–226, 2002.

[13] Balázs Bank, "Accurate and efficient modeling of beating and two-stage decay for string instrument synthesis," in *Proc. MOSART Workshop on Curr. Res. Dir. in Computer Music*, Barcelona, Spain, November 2001, pp. 134–137.