An Efficient Time–Frequency Method for Synthesizing Noisy Sounds With Short Transients and Narrow Spectral Components

Damián Marelli, Mitsuko Aramaki,
Richard Kronland-Martinet, and Charles Verron

***Abstract*—The inverse fast Fourier transform (IFFT) method is a time–frequency technique which was proposed to alleviate the complexity of the additive sound synthesis method in real-time applications. However, its application is limited by its inherent tradeoff between time and frequency resolutions, which are determined by the number of frequencies used for time–frequency processing. In a previous work, the authors proposed a frequency-refining technique for overcoming this frequency limitation, permitting achieving any time and frequency resolution using a small number of frequencies. In this correspondence we extend this work, by proposing a time-refining technique which permits overcoming the time resolution limitation for a given number of frequencies. Additionally, we propose an alternative to the frequency-refining technique proposed in our previous work, which requires about half the computations. The combination of these two results permits achieving any time and frequency resolution for any given number of frequencies. Using this property, we find the number of frequencies which minimizes the overall complexity. We do so considering two different application scenarios (i.e., offline sound design and online real-time synthesis). This results in a major complexity reduction in comparison with the design proposed in our previous work.**

***Index Terms*—Colored noisy signal, inverse fast Fourier transform (IFFT) sound synthesis, short transient signal.**

I. Introduction

SYNTHESIS of environmental sounds has received an increasing interest in the last decade for virtual reality, animation and physically based simulations. Several models have been proposed for simulating liquid sounds [1]–[3], contact sounds (such as impact [4], fracture [5], deformation and friction sounds [6]), and aerodynamic phenomena (like wind and fire) [7]–[9]. The synthesis techniques can be broadly classified into two main categories. The first category is based on physical models, which aim at simulating the physics of sound sources [10]. The second category is based on signal models, which aim at reproducing perceptual effects independently of the physics of the source [11]. Some signal models also involve physically inspired control parameters (like the speed of the wind, the size of a resonating object...) leading to a hybrid class of "physically inspired signal models" [8], [12], [13].

Among linear signal models, the "sinusoids plus noise" model [14] has been successfully applied for analysis, transformation and synthesis of a wide class of environmental sounds [15], [16]. The model represents a sound as a combination of a deterministic contribution (i.e., a sum of time-varying sinusoids, also called partials) and a stochastic contribution (i.e., a "time-varying filter" with white noise at its input). One great advantage of this model is that the synthesis parameters can be determined from the analysis of natural sounds. Also, modifications of the synthesis parameters lead to high-quality parametric transformations, such as pitch-shifting, time-stretching and morphing. These transformations are often performed "offline." In this scenario, the synthesis parameters are precomputed and stored, e.g., in a sound description interchange format (SDIF) file [17]. However, some interactive synthesis scenarios involve real-time ("online") manipulation of the synthesis parameters. The complexity requirements assigned to sound synthesis depends on the application. Non-real-time applications may afford costly synthesis models, but real-time applications typically impose heavy constraints on the computational cost. To address this issue, a computationally efficient method based on "inverse fast Fourier transform" (IFFT) was proposed for synthesizing the deterministic and stochastic contributions of the "sinusoids plus noise" model [14], [18]. This method was used for designing musical and environmental sound synthesizers satisfying real-time constraints [16], [19].

The IFFT method requires a number of frequencies (which equals the synthesis window size) matching the length of the auto-correlation function of the sound to be synthesized. Thus, a large number of frequencies is needed to reproduce narrowband noisy components. However, this is not compatible with the generation of short transient signals. This is an important issue when synthesizing environmental sounds, which require the generation of deterministic and stochastic signals with a wide range of temporal and spectral behaviors. For instance, sounds such as impacts [20], drops of water [1] or fire cracklings [8] have very sharp transients. On the other hand, sounds such as wind whistling or fire hissing have very narrowband noisy components [8]. An approach for synthesizing these different sorts of environmental sounds with the IFFT method was proposed in [16] by using several synthesis window sizes in parallel. A drawback of this approach is that it cannot synthesize narrow noisy components having short transients. This limitation was overcome by the method proposed by the authors in [21], where a small number of frequencies was used to guarantee the synthesis of short transients. Then, a frequency-refining technique is used to go around the resulting frequency resolution limitation. This technique consists in generating time–frequency noise with an auto-correlation function such that the noise obtained after converting it to time domain has the desired power spectral density. This method has the potential for synthesizing filtered noise with an extended range of time–frequency properties, in comparison to IFFT synthesis. However, this important property comes at the expense of an increased computational complexity, which limits its application in real-time synthesis applications.

In this correspondence, we extend our work [21] in two ways. First, we propose a time-refining technique which permits synthesizing short transients (with a time resolution smaller than the length of the synthesis window) while using a large number of frequencies. This technique can be seen as the dual of the frequency-refining technique proposed in [21], in the sense that it permits overcoming the time resolution limitation. Second, we propose an alternative to that frequency-refining technique. More precisely, in order to generate time–frequency noise, the technique in [21] starts with white noise in the time domain, and uses an analysis stage (filterbank) to convert it to the time–frequency domain. Instead, the proposed technique generates the desired time–frequency noise without using an analysis stage, significantly reducing the overall complexity. Hence, combining these two techniques, any time and frequency resolution can be achieved by carrying out the

Manuscript received May 21, 2010; revised June 30, 2011 and October 17, 2011; accepted October 18, 2011. Date of publication November 16, 2011; date of current version March 02, 2012. This work was supported in part by the project "senSons" from the French National Research Agency (ANR). The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Sylvain Marchand.

D. Marelli is with the School of Electrical Engineering and Computer Science, University of Newcastle, Callaghan, NSW 2308, Australia (e-mail: damian.marelli@newcastle.edu.au).

M. Aramaki is with the CNRS-Institut de Neurosciences Cognitives de la Méditerranée, , 13402 Marseille Cedex 20, France (e-mail: aramaki@incm.cnrs-mrs.fr).

R. Kronland-Martinet and C. Verron are with the CNRS-Laboratoire de Mécanique et d'Acoustique, 13402 Marseille Cedex 20, France (e-mail: kronland@lma.cnrs-mrs.fr; charles.verron@gmail.com).

Digital Object Identifier 10.1109/TASL.2011.2176334

synthesis in the time–frequency domain using any number of frequencies. Using this, we study which is the optimal number of frequencies in the sense of minimizing the overall complexity. We do so considering the two scenarios described above, depending on whether the synthesis parameters are computed offline or online. In both cases, the resulting optimal configuration leads to a significant complexity reduction when compared with the synthesis method proposed in [21].

As we point out in Section II, the IFFT synthesis method is equivalent to a synthesis filterbank architecture. A synthesis technique related to this architecture is the *phase vocoder* [22]. As pointed out in [22], this technique is equivalent to a filterbank, with the only difference in that each frequency band is multiplied by a complex modulating term, which is not present in the filterbank architecture. Hence, the proposed techniques are readily applicable to carry out sound synthesis using the phase vocoder architecture, by considering the extra complex modulating term in the design.

The rest of the correspondence is organized as follows. In Section II, we briefly introduce the IFFT synthesis method, which states the basis for our proposed synthesis method. In Section III, we describe the frequency-and time-refining techniques used to overcome the frequency and time resolutions of the IFFT method, and we combine them to propose a unified time–frequency synthesis method. In Section IV, we address the synthesizer design, including the optimal choice for the number of frequencies. In Section V, we illustrate the application of the proposed scheme by synthesizing realistic environmental sounds, and we also compare the proposed frequency-refining technique with the one proposed in [21]. Finally, we give concluding remarks in Section VI.

Notation: Throughout the correspondence, we will use the following notational convention: scalars are denoted using normal (i.e., non-bold) lowercase letters (e.g., x); and vector and matrix using lowercase bold letters (e.g., $\mathbf{x}$) and uppercase bold letters (e.g., $\mathbf{X}$), respectively. The ith entry of a vector $\mathbf{x}$ is denoted by $[\mathbf{x}]_i$ and the i,jth entry of a matrix $\mathbf{X}$ is denoted by $[\mathbf{X}]_{i,j}$. Finally, $\overline{\mathbf{X}}$ denotes the complex conjugate of the matrix $\mathbf{X}$ and $\mathbf{X}^*$ denotes its transpose conjugate i.e., $\mathbf{X}^* = \overline{\mathbf{X}^T}$.

II. Noisy Sound Synthesis Using the IFFT Method

A noisy sound $y(t)$ is generally modeled as a stochastic process with a time-varying spectrum

$$\phi_y(z,t) = \mathcal{Z}_1\left\{r_y(\tau,t)\right\}$$

where $\mathcal{Z}_1\{\cdot\}$ denotes the z-transform with respect to the first variable, and

$$r_y(\tau,t) = \mathcal{E}\left\{y(t)y(t-\tau)\right\}$$

with $\mathcal{E}\{\cdot\}$ denoting expected value [14]. Using the IFFT method, the sound $y(t)$ is synthesized by the following overlap-add procedure:

$$y(t) = \sum_{k=-\infty}^{\infty} f(t-kD)v^{(k)}(t-kD) \tag{1}$$

where $f(t), t \in \mathbb{Z}$ is a synthesis window of size M (i.e., $f(t)=0$ if $t<0$ or $t \geq M$), which is assumed to satisfy

$$\sum_{\tau=-\infty}^{\infty} f^2(t-\tau D) = 1 \quad \text{for all} \quad t \in \mathbb{Z} \tag{2}$$

to conserve energy, and $D \leq M$ is the synthesis hop size. The kth block of M samples $v^{(k)}(t), t=0,\cdots,M-1$ is obtained doing the inverse discrete Fourier transform (DFT) of an M-dimensional random vector $\mathbf{v}(k)$, i.e.,

$$v^{(k)}(t) = \sum_{m=1}^{M} \left[\mathbf{v}(k)\right]_m e^{j2\pi\frac{m-1}{M}t}$$

where

$$\left[\mathbf{v}(k)\right]_m = \phi_y\left(e^{j2\pi\frac{m-1}{M}}, kD\right)\left[\mathbf{w}(k)\right]_m \tag{3}$$

with $\mathbf{w}(k)$ being a white complex vector random process (i.e., a sequence of uncorrelated complex random vectors with uncorrelated entries) with Gaussian distribution.

As shown in [21], (1) is equivalent to a synthesis filterbank operation, i.e.,

$$y(z) = \sum_{m=1}^{M} f_m(z) \uparrow_D \left\{[\mathbf{v}]_m\right\}(z) \tag{4}$$

where the filters $f_m(z) = f(e^{j2\pi(m-1)/M}z)$, $m = 1,\cdots,M$ are frequency-shifted versions of the synthesis window $f(z)$, and $\uparrow_D \{[\mathbf{v}]_m\}(k)$ denotes the upsampling operation with factor D (i.e., inserting $D-1$ zeros between every two samples) applied to the signal $[\mathbf{v}(k)]_m$.

It follows from (4) that the frequency resolution of the IFFT method is determined by the spectral shape of the synthesis window $f(t)$, and that its time-resolution is given by its time domain concentration. Hence, this method suffers from an inherent tradeoff between time and frequency resolution, turning the synthesis of narrowband noises incompatible with the generation of short transient signals.

III. Overcoming Time and Frequency Resolution Limitations

In this section, we propose two techniques which permit overcoming the aforementioned tradeoff. In Sections III-A and III-B, we explain how to achieve arbitrary frequency and time resolutions, respectively, and in Section III-C we explain how to combine these two techniques to build a time–frequency synthesizer without time and frequency resolution limitations.

A. Achieving Arbitrary Frequency Resolution

In this section, we assume that we want to synthesize a stationary random process with an arbitrary spectral shape, which may include components narrower than the frequency concentration of the synthesis window $f(t)$. A method for doing so was proposed in [21]. That method starts by generating scalar white noise which is converted to the time–frequency domain using an analysis filterbank. The resulting signal is then processed by a transfer matrix and its output is converted back to time domain using a synthesis filterbank. In this section, we propose an alternative strategy for carrying out the same task. The essential difference is that the proposed strategy does not need the analysis filterbank stage. Instead, we design the transfer matrix so that vector white noise is directly applied to its input. The advantage of doing so is that this noise can be real-valued, hence the complexity associated with the transfer matrix is reduced by two.

As depicted in Fig. 1, the idea consists of processing an N-dimensional white random vector $\mathbf{w}(k)$ (the value of N is determined by the number of columns of the transfer matrix $\mathbf{R}(z)$ in (6) below) by an $M \times N$ transfer matrix $\mathbf{S}(z)$ so that the signal $y(t)$ obtained after synthesis has the desired power spectrum.

Remark 1: Notice that the scheme in Fig. 1 can be interpreted as a generalization of the IFFT method described in Section II. More precisely, the IFFT scheme is obtained by replacing the $M \times N$

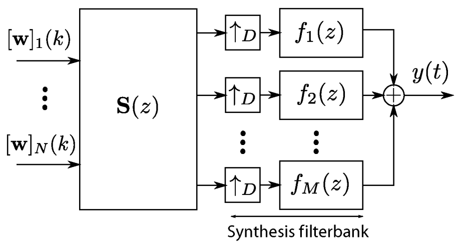

Fig. 1. Scheme for achieving an arbitrary spectral shape.

transfer matrix $\mathbf{S}(z)$ by a diagonal $M \times M$ static matrix $\mathbf{S}$, with entries $[\mathbf{S}]_{m,m} = \phi_y(e^{j2\pi(m-1)/M}, kD)$, $m = 1, \cdots, M$. Also, in the IFFT method, the synthesis window $f(z)$ needs to be chosen so that its impulse response has length M and satisfies (2).

Using polyphase representation [23], we can write

$$\mathbf{y}(z) = \mathbf{F}(z)\mathbf{S}(z)\mathbf{w}(z)$$

where $\mathbf{y}(z)$ and $\mathbf{F}(z)$ are the polyphase representations of $y(t)$ and the synthesis filterbank, respectively, having impulse responses

$$[\mathbf{y}(k)]_d = y(kD + 1 - d),$$
$$[\mathbf{F}(k)]_{d,m} = f_m(kD - d + 1)$$

for all $m = 1, \cdots, M$, and $d = 1, \cdots, D$. Let $\phi_{\mathbf{y}}(z) = \mathcal{E}\{\mathbf{y}(z)\mathbf{y}^*(z)\}$ be the spectrum of $\mathbf{y}(z)$. We have that

$$\begin{aligned}\phi_{\mathbf{y}}(z) &= \mathbf{F}(z)\mathbf{S}(z)\mathcal{E}\{\mathbf{w}(z)\mathbf{w}^*(z)\}\mathbf{S}^*(z)\mathbf{F}^*(z)\\ &= \mathbf{F}(z)\mathbf{S}(z)\mathbf{S}^*(z)\mathbf{F}^*(z).\end{aligned} \tag{5}$$

Let $\phi_{\mathbf{y}}(z) = \mathbf{R}(z)\mathbf{R}^*(z)$ be a spectral factorization of $\phi_{\mathbf{y}}(z)$ [24]. Then, from (5), the required matrix $\mathbf{S}(z)$ needs to satisfy

$$\mathbf{R}(z) = \mathbf{F}(z)\mathbf{S}(z). \tag{6}$$

The minimum-norm solution of (6) is given by:

$$\mathbf{S}(z) = \mathbf{F}^\dagger(z)\mathbf{R}(z) \tag{7}$$

where $\mathbf{F}^\dagger(z)$ is the Moore-Penrose pseudoinverse [25] of $\mathbf{F}(z)$, which is shown to be equal to the polyphase representation $\tilde{\mathbf{F}}(z)$ of the dual window $\tilde{f}(z)$ [26] of $f(z)$. While (7) can be easily computed, it produces a transfer matrix $\mathbf{S}(z)$ whose impulse response $\mathbf{S}(t)$ has a very large number of nonzero entries. Hence, its implementation is very inefficient. To improve efficiency, the number of nonzero entries in $\mathbf{S}(t)$ can be reduced by solving (6) using sparse approximation techniques [27]. Generally speaking, these techniques aim at (approximately) solving the following minimization problem:

$$\mathbf{S}(z) = \underset{\mathbf{S}(z):\#\{\mathbf{S}(t)\}\le l}{\arg\min} \|\mathbf{R}(z) - \mathbf{F}(z)\mathbf{S}(z)\| \tag{8}$$

where, for any $D \times D$ transfer matrix $\mathbf{X}(z)$, $\|\mathbf{X}(z)\| = \langle \mathbf{X}(z), \mathbf{X}(z)\rangle^{1/2}$, with

$$\langle \mathbf{X}(z), \mathbf{Y}(z)\rangle = \frac{1}{2\pi D}\int_{-\pi}^{\pi} \mathrm{Tr}\left\{\mathbf{X}(e^{j\omega})\mathbf{Y}^*(e^{j\omega})\right\} d\omega \tag{9}$$

and $\mathbf{S}(z) : \#\{\mathbf{S}(t)\} \le l$ means that the minimization is done over all matrices whose impulse response has at most l nonzero entries. To solve (8) we use the orthogonal matching pursuit (OMP) algorithm [28], [29]. We describe the resulting algorithm below.

For each $m, n = 1, \cdots, M$ and $\tau \in \mathbb{Z}$, define the $D \times D$ transfer matrix

$$\mathbf{V}_{m,n,\tau}(z) = \mathbf{F}(z)\mathbf{U}_{m,n,\tau}(z) \tag{10}$$

where the $M \times N$ impulse response of $\mathbf{U}_{m,n,\tau}(z)$ is defined by

$$[\mathbf{U}_{m,n,\tau}(t)]_{i,j} = \begin{cases} 1, & i = m, j = n, t = \tau \\ 0, & \text{otherwise.}\end{cases}$$

Then, $\mathbf{S}(z)$ is computed using the following iterative procedure: Let $\hat{\mathbf{R}}^{(0)}(z) = 0$. Then, at iteration k we compute

$$\begin{aligned}&\left(m^{(k)}, n^{(k)}, \tau^{(k)}\right)\\ &\quad = \underset{(m,n,\tau)}{\arg\max} \frac{\left|\left\langle \mathbf{R}(z) - \hat{\mathbf{R}}^{(k-1)}(z), \mathbf{V}_{m,n,\tau}(z)\right\rangle\right|}{\|\mathbf{V}_{m,n,\tau}(z)\|} && (11)\\ &\hat{\mathbf{R}}^{(k)}(z)\\ &\quad = \mathbf{F}(z)\hat{\mathbf{S}}^{(k)}(z) && (12)\\ &\hat{\mathbf{S}}^{(k)}(z)\\ &\quad = \underset{\mathrm{supp}\{\mathbf{S}(t)\}\subseteq\mathcal{S}^{(k)}}{\arg\min} \|\mathbf{R}(z) - \mathbf{F}(z)\mathbf{S}(z)\| && (13)\end{aligned}$$

where $\mathcal{S}^{(k)}$ is the set that includes the indexes $\{(m_l, n_l, \tau_l) : l = 1, \cdots, k\}$, and $\mathrm{supp}\{\mathbf{S}(t)\} \subseteq \mathcal{S}^{(k)}$ means that the minimization with respect to $\mathbf{S}(z)$ is done over all matrices whose impulse responses have zeros outside $\mathcal{S}^{(k)}$.

Remark 2: It is straightforward to verify that

$$\begin{aligned}&\left\langle \mathbf{R}(z) - \hat{\mathbf{R}}^{(k-1)}(z), \mathbf{V}_{m,n,\tau}(z)\right\rangle\\ &\quad = \left\langle \left(\mathbf{R}(z) - \hat{\mathbf{R}}^{(k-1)}(z)\right), \mathbf{F}(z)\mathbf{U}_{m,n,\tau}(z)\right\rangle\\ &\quad = \left\langle \mathbf{F}^*(z)\left(\mathbf{R}(z) - \hat{\mathbf{R}}^{(k-1)}(z)\right), \mathbf{U}_{m,n,\tau}(z)\right\rangle\\ &\quad = \mathcal{Z}^{-1}\left\{\left[\mathbf{F}^*(z) * \left(\mathbf{R}(z) - \hat{\mathbf{R}}^{(k-1)}(z)\right)\right]_{m,n}\right\}(\tau)\end{aligned}.$$

Hence, the computation of all the inner product in (11) can be done by computing the impulse response of $\mathbf{F}^*(z)(\mathbf{R}(z) - \hat{\mathbf{R}}^{(k-1)}(z))$. Also, the minimization in (13) is a linear least squares problem whose solution can be easily computed.

B. Achieving Arbitrary Time Resolution

Now suppose that we want to change the amplitude of the spectrum of the random process synthesized in Section III-A, following an arbitrary law $a(t)$, which may include transients shorter than the time concentration of the synthesis window $f(t)$. We can do this in the time domain by multiplying by $a(t)$ the output of the synthesis filterbank $f_m(z)$. To transpose this operation to the time–frequency domain, we add after $a(t)$ a perfect-reconstructing pair of analysis and synthesis filterbanks (i.e., a pair which achieves perfect reconstruction), as shown in Fig. 2.

The analysis filterbank operation consists of filtering $y(t)$ using an array of filters $h_m(z)$, $m = 1, \cdots, M$ followed by a downsampling operation with factor D (i.e., by keeping one out of D samples). Then, as shown in the Appendix, the impulse response $[\mathbf{T}]_{m,n}(l, k)$ of the m, nth entry of the $M \times M$ transfer matrix $\mathbf{T}(z, k)$ shown in Fig. 2 is given by

$$[\mathbf{T}]_{m,n}(l, k) = \left(\left(a^{(k)}h_m\right) * f_n\right)(lD) \tag{14}$$

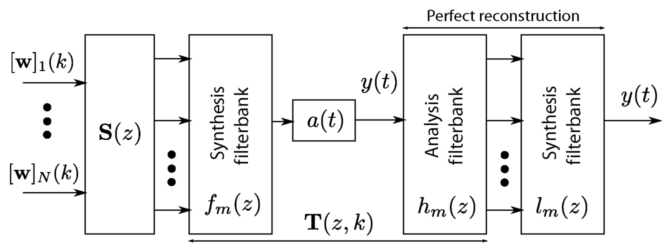

Fig. 2. Scheme for achieving arbitrary time resolution.

where $a^{(k)}(t) = a(kD - t)$. Let C be the impulse response length of $h_m(z)$, $m = 1, \cdots, M$. For each k, we can expand $a^{(k)}(t)$, $t = 0, \cdots, C-1$ using DFT as follows:

$$a^{(k)}(t) = \sum_{c=1}^{C} \alpha^{(c)}(k) e^{j2\pi \frac{c-1}{C} t}. \tag{15}$$

Then, as shown in the Appendix, we have that

$$\mathbf{T}(l,k) = \sum_{c=1}^{C} \alpha^{(c)}(k) \mathbf{T}^{(c)}(l) \tag{16}$$

where the impulse response $[\mathbf{T}^{(c)}]_{m,n}(l)$ of the m,nth entry of $\mathbf{T}^{(c)}(z)$ is given by

$$\left[\mathbf{T}^{(c)}\right]_{m,n}(l) = \left(h_m^{(c)} * f_n\right)(lD) \tag{17}$$

$$h_m^{(c)}(t) = h_m(t) e^{j2\pi \frac{c-1}{C} t} \tag{18}$$

where $*$ denotes convolution.

Remark 3: Notice that (16) permits expressing the time-varying transfer matrix $\mathbf{T}(l,k)$ as a time-varying weighted sum of the time-invariant transfer matrices $\mathbf{T}^{(c)}(l)$.

Remark 4: Since the filters $h_m(z)$, $m = 1, \cdots, M$ are frequency-shifted versions of the same prototype $h(z)$, if $R = C/M$ is integer, then $h_m^{(c+rR)} = h_{m+r}^{(c)}$, and therefore, the output of $\mathbf{T}_{m,n}^{(c+rR)}$ is obtained from that of $\mathbf{T}_{m,n}^{(c)}$ by doing an r-step circular shift. Also, $h_m^{(c+C-2)} = \overline{h_m^{(c)}}$, and therefore, the output of $\mathbf{T}_{m,n}^{(c+C-2)}$ is obtained from that of $\mathbf{T}_{m,n}^{(c)}$ by complex conjugation. Hence, only the $\lceil R/2 \rceil$ ($\lceil x \rceil$ denotes the smallest integer greater than or equal to x) transfer matrices $\mathbf{T}_{m,n}^{(c)}(l)$, $c = 1, \cdots, \lceil R/2 \rceil$ need to be computed to generate (16).

C. Time–Frequency Synthesizer Architecture Without Time and Frequency Resolution Limitations

Using the results above we can design a time–frequency method for synthesizing noisy sounds with arbitrary frequency and time resolutions. The idea is to use the method described in Section III-A to generate a number P of contiguous narrow frequency components, the amplitude of each of which is modified using the method explained in Section III. The scheme is depicted in Fig. 3. The number P of frequency bands is chosen to achieve the desired frequency resolution. Notice that P does not need to be equal to the number M of frequency bands used in the time–frequency representation. Then, for each $p = 1, \cdots, P$, a white, N_p-dimensional vector random process $\mathbf{w}_p(k)$ is applied to the input of an $M \times N_p$ transfer matrix $\mathbf{S}_p(z)$. The matrix $\mathbf{S}_p(z)$ is designed as described in Section III-A, so that it generates at the output of the synthesis filterbank $l_m(z)$, a narrow frequency band with the desired spectral shape. Then, for each $k \in \mathbb{Z}$,

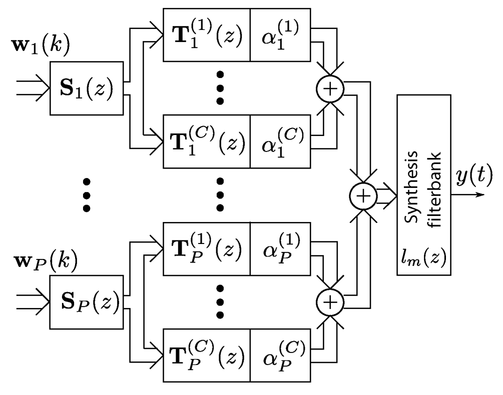

Fig. 3. Time–frequency synthesis architecture.

the coefficients $\alpha_p^{(c)}(k)$, $c = 1, \cdots, C$ are computed from the desired time-varying amplitude $a_p(t)$ of the pth frequency band using (15) (or the more efficient method described in Section IV-B). The constants $\alpha_p^{(c)}(k)$ are then used to multiply the output of the $M \times M$ transfer matrices $\mathbf{T}^{(c)}(z)$, $c = 1, \cdots, C$, which are computed as explained in Section III-B.

Depending on the application, the computation of the coefficients $\alpha_p^{(c)}(k)$, $c = 1, \cdots, C$ can be performed either online or offline. We refer to these two scenarios as *online synthesis* and *offline synthesis*, respectively. Note that in both scenarios, the sound synthesis is performed in real-time. The only difference is that in the online scenario the time-varying amplitudes $a_p(t)$ are specified "on the fly", while in the offline scenario these amplitudes are known in advance (e.g., stored in an SDIF file).

Remark 5: The scheme in Fig. 3 requires the computation of P transfer matrices of dimensions $M \times N_p$, and $P \times \lceil R/2 \rceil$ transfer matrices of dimension $M \times M$ (recall Remark 4). Hence, it seems to suffer from a very high complexity. However, notice that the nonzero entries of the matrices $\mathbf{S}_p(z)$ concentrate in a neighborhood of the row corresponding to the center of the pth frequency band. Also, in view of (17), the nonzero entries of the matrices $\mathbf{T}^{(c)}(z)$ concentrate towards its main diagonal. Hence, the composition $\mathbf{T}^{(c)}(z)\mathbf{S}_p(z)$ can be very efficiently implemented.

IV. Synthesizer Design

The IFFT method described in Section II carries out the synthesis operation using M frequency bands, and its maximum time and frequency resolutions are determined by this value. As explained in Section III-C, the proposed synthesis architecture overcomes these resolution limitations using time- and frequency-refining techniques. Hence, any time and frequency resolution can be achieved using any value of M. In this section, we use this architecture to design a noisy sound synthesizer. We use a sampling frequency of $f_s = 44.1$ kHz. We want a frequency resolution of $P = 1024$ frequencies over the range $[-f_s/2, f_s/2]$, and we want the amplitude of the signal in each frequency band to vary once every $L = 64$ samples, i.e., 1.5 ms.

In Sections IV and IV-B, we address the synthesizer design, assuming that the values of M and (the downsampling factor) D are given. In Section IV-C, we use this design to study the optimal values of M and D (in the sense of minimizing the overall complexity) for both, online and offline synthesis. The resulting optimal offline synthesis scheme slightly differs from the one depicted in Fig. 3, and is therefore summarized in Section IV-D. Finally, in Section IV-E we compare the proposed synthesis schemes with the one proposed in [21].

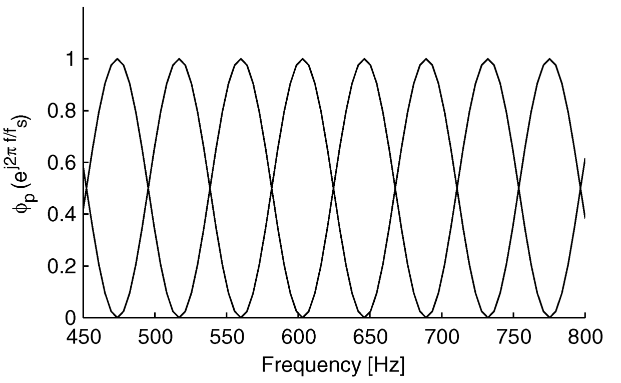

Fig. 4. Shape of the individual spectral components $\phi_p(z)$.

A. Synthesizer Design for Given Values of M and D

In this section, we address the synthesizer design. To this end, we assume that the input data for the design is: the number P of frequency channels (determining the frequency resolution), the rate L (in samples) with which the amplitudes of each frequency channel are specified (determining the time resolution), the sequences $b_p(t)$, $p = 1, \cdots, P$, defining the amplitudes of each frequency channel, specified once every L samples, the number M of frequency bands and the downsampling factor D.

For each $p = 1, \cdots, P$, we design the spectral shaping transfer matrix $\mathbf{S}_p(z)$ so that it generates, at the output, a signal with spectrum $\phi_p(z) = |g_p(z)|^2$, where $g_p(z)$ is a root raised cosine filter [30] with roll-off factor $\beta = 1$, bandwidth $f_g = 2f_s/P = 86$ Hz and center frequency $f_p = (p-1)f_s/P$

$$g_p(e^{j2\pi f/f_s}) = \begin{cases} \sqrt{\frac{1}{2}\left(1+\cos\frac{2\pi(f-f_p)}{f_g}\right)}, & |f-f_p| < \frac{f_g}{2} \\ 0, & \text{otherwise.} \end{cases} \quad (19)$$

Some of the resulting spectral shapes are shown in Fig. 4. Notice that this design guarantees that

$$\sum_{p=1}^{P} \phi_p(e^{j\omega}) = 1 \quad \text{for all} \quad \omega \in [-\pi, \pi]$$

hence a flat spectrum is obtained when all bands have the same amplitude. Each $\mathbf{S}_p(z)$ is designed using the iterative procedure (11)–(13), which is stopped when the relative energy difference between $\phi_p(z)$ and the spectrum induced by $\mathbf{S}_p(z)$ is smaller than -40 dB.

It follows from equation (17) that the filters $f_m(z), h_m(z)$, $m = 1, \cdots, M$ need to be concentrated in frequency, so that the off-diagonal terms of $\mathbf{T}^{(c)}(z)$, $c = 1, \cdots, C$ vanish quickly. However, if their frequency response is too concentrated, the impulse response length C (of $h_m(t)$, $m = 1, \cdots, M$) becomes too big, and therefore, a large number $\lceil R/2 \rceil$ of transfer matrices $\mathbf{T}^{(c)}(z)$ need to be computed (recall Remark 4). We found a good compromise by choosing $R = 3$ and designing the prototypes $f(z)$ and $h(z)$ as FIR filters having impulse response length $C = MR$, which best approximate, in a least-squares sense, a raised cosine window with roll-off factor $\beta = 1$, bandwidth $f_f = f_s/M$ and center frequency 0, i.e.,

$$f(e^{j2\pi f/f_s}) = \begin{cases} \frac{1}{2}\left(1+\cos\frac{2\pi f}{f_f}\right), & |f| < \frac{f_f}{2} \\ 0, & \text{otherwise} \end{cases} \quad (20)$$

and $h(z) = f(z)$. As mentioned in Section III-B, the prototype $l(z)$ of the synthesis filters needs to be designed so that its output is an approximate reconstruction of the input to the filters $h_m(z), m = 1, \cdots M$. We design $l(z)$ using the iterative algorithm in [31, Eq. (5.3.16)]. The iterations are stopped when the relative analysis/synthesis reconstruction error becomes smaller than -40 dB. The resulting impulse response $l(t)$ is then truncated by removing the leading and trailing taps which are 60 dB smaller than the maximum absolute value.

The transfer matrices $\mathbf{T}^{(c)}(z)$, $c = 1, \cdots, \lceil R/2 \rceil$ are computed using (17). In order to reduce the implementation complexity, we truncate them by zero-rounding their entries having absolute values smaller than a threshold chosen 40 dB smaller than the maximum absolute value.

As mentioned before, the amplitude of each frequency band is specified once every $L = 64$ samples. In order to build the coefficients $\alpha_p^{(c)}(k)$, $p = 1, \cdots, P$, $c = 1, \cdots, C$ (using either (15) or the efficient method described in Section IV-B), we need to interpolate these values to obtain one value per sample. We do so using a Hann window [32] of length $2L+1$, which is concentrated in frequency and hence minimizes the number of terms required in the expansion (16). The interpolated amplitude function has bandwidth f_s/L, and therefore only $1 + 2C/L$ terms need to be considered in the expansion (16). Moreover, only $\bar{C} = 1 + C/L$ need to be computed since the remaining terms are obtained by complex conjugation.

B. Computation of $\alpha^{(c)}(k)$

For each frequency channel $p = 1, \cdots, P$, equation (15) indicates that the synthesis parameters $\alpha_p^{(c)}(k)$, $c = 1, \cdots, C$ are computed using DFT on the segment $a_p(t), kL \leq t \leq kL + C - 1$ of the time-varying amplitude signal $a_p(t)$. This requires $2C\log_2(C)/D$ multiplications per sample and per frequency component present in the sound to be synthesized. As mentioned in Section III-C, this computation can be carried out either online or offline. In the former case, its associated complexity is a critical issue. Since $a_p(t)$ is obtained from an interpolation procedure, and only $\bar{C}$ coefficients $\alpha_p^{(c)}(k)$ need to be computed, for each p and each k (recall the last paragraph of Section IV-A), it turns out that there is a more efficient method to carry out this computation. We describe this method next.

Recall that $b_p(t)$ denotes the sequence of amplitudes for the frequency channel p, specified once every L samples. Let $e(t)$ denote the Hann window used for interpolation, i.e., $a_p(t) = (e * \uparrow_L \{b_p\})(t)$ (recall that $\uparrow_L \{\cdot\}$ denotes the upsampling operation with factor L). Now, it is straightforward to see that the segmented DFT operation on $a_p(t)$ is equivalent to an analysis filterbank operation. This filterbank is formed by the filters $r_c(z)$, $c = 1, \cdots, C$, derived from the prototype

$$r(t) = \begin{cases} 1, & 0 \leq t \leq C-1 \\ 0, & \text{otherwise} \end{cases}$$

by inverse frequency modulation (i.e., $r_c(z) = r(e^{-j2\pi(c-1)/C}z), c = 1, \cdots, C$), followed by a downsampling operation with factor D. Let $p_c(z) = e(z)r_c(z), c = 1, \cdots, C$. Then, for each $c = 1, \cdots, C$, the sequence $\alpha_p^{(c)}(k)$ is given by

$$\alpha_p^{(c)}(k) = \downarrow_D \{p_c * \uparrow_L \{b_p\}\}(t). \quad (21)$$

Since only $\bar{C}$ coefficients need to be computed, for each frequency component in the sound to be synthesized, (21) requires $2(2L+C)\bar{C}/(LD)$ multiplications per sample.

To see the computational advantages of the proposed method, we consider the case where $M = 256$ and $D = 224$, as in the optimal online synthesis design described in Section IV-D. Using the DFT procedure in (15), the computation of the synthesis parameters $\alpha_p^{(c)}(k)$ requires 66 multiplications per sample and per frequency component, while using (21) requires 1 multiplication.

C. Optimal Values of M and D

The designs presented in Sections IV-A and IV-B assume that M and D are given. A natural question is how to choose these two values

TABLE I
OPTIMAL DOWNSAMPLING FACTOR D FOR EACH NUMBER M OF FREQUENCY BANDS, CONSIDERING BOTH, ONLINE AND OFFLINE SYNTHESIS

M	32	64	128	256	512	1024
D (off-line)	28	52	108	222	367	996
D (on-line)	28	52	108	224	458	1018

so that the overall complexity of the synthesis algorithm is minimized. In regard to the value of M, an extreme case consists in choosing it small enough so that the desired time resolution is readily achievable without using time-refining. Then, the desired frequency resolution is achieved using frequency-refining. This is the configuration used in [21]. In the other extreme case, M is chosen large enough so that the desired frequency resolution is readily achievable, and the desired time resolution is achieved using time-refining. In this section we study the optimal choice of M between these two extreme cases.

For each value of M, the downsampling factor D needs to satisfy $1 \leq D \leq M$. While a small value of D requires that the branches before the synthesis filterbank $l_m(z)$ in Fig. 3 are computed more often, a large value of D increases the impulse response length of the synthesis filters $l_m(z), m = 1, \cdots, M$, as well as the number of nonzeros entries in the impulse responses of the transfer matrices $\mathbf{S}_p(z)$. In the case of online synthesis, it also affect the computation of the synthesis coefficients $\alpha_p^{(c)}(k)$, as described in Section IV-B. Hence, for each M, the optimal value of D needs to be determined by a numerical search.

To measure the complexity of the synthesis algorithm, for given M and D, we count the number of real multiplications per synthesized sample. To do so, notice that in Fig. 3, only $P/2$ branches need to be computed since the synthesized signal $y(t)$ is real valued. Also, it follows from (5) that we can choose the white vector random processes $\mathbf{w}_p(k)$, $p = 1, \cdots, P/2$ to be real. After doing so, the computation of each $\mathbf{S}_p(z)$ requires $2\#\{\mathbf{S}(t)\}/D$ multiplications (recall that $\#\{\mathbf{S}(t)\}$ denotes the number of nonzero entries in the impulse response $\mathbf{S}_p(t)$). On each branch in Fig. 3, the computation of each $\mathbf{T}^{(c)}(z)$, $c = 1, \cdots, \lceil R/2 \rceil$ requires $4\#\{\mathbf{T}^{(c)}(t)\}/D$ multiplications, and the multiplication with the constants $\alpha_P^{(c)}(k)$ requires $4\bar{C}/D\#\{\text{rows}\{\mathbf{T}^{(c)}(z)\mathbf{S}_p(z)\}\}$ multiplications, where $\#\{\text{rows}\{\mathbf{T}^{(c)}(z)\mathbf{S}_p(z)\}\}$ denotes the number of nonzero rows of the transfer matrix $\mathbf{T}^{(c)}(z)\mathbf{S}_p(z)$. Finally, using the algorithm in [33], and assuming that M is a power of two, so that an M-point FFT can be implemented with $2M \log_2 M$ (real) multiplications using the Radix-2 algorithm [34, Ch. 6.1], the implementation of the synthesis filterbank $l_m(z)$ requires $(Q + 2M \log_2 M)/D$ multiplications, where Q denotes the impulse response length of the synthesis filter prototype $l(z)$.

Since M needs to be a power of two, our search is constrained to $M \in \{32, 64, 128, 256, 512, 1024\}$. Using the above, for each value of M, we choose the value of D that minimizes the overall computation. We do so for both, offline and online synthesis. For the case of online synthesis, we consider the extreme case in which the sound to be synthesized has $P/2 = 512$ frequency components (notice that in general, a sound has fewer components, e.g., the glass impact studied in Section V-A). The optimal values of D are shown in Table I.

In Figs. 5 and 6, we show the computational cost of the offline and the online synthesis algorithms, respectively, resulting from each value of M, and using the values of D from Table I. We show the total computational cost, as well as that of each stage, i.e., the frequency-refining stage described in Section III-A, the time-refining stage described in Section III-B, the computation of the synthesis parameters $\alpha_p^{(c)}(k)$ (only for the online synthesis method), and the synthesis filterbank

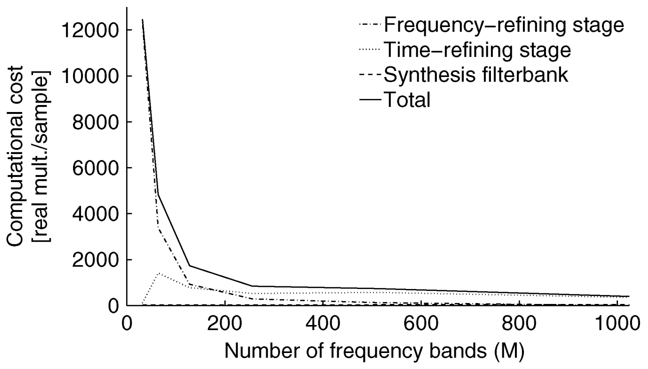

Fig. 5. Computational cost of the offline synthesis algorithm versus the number of frequency bands M.

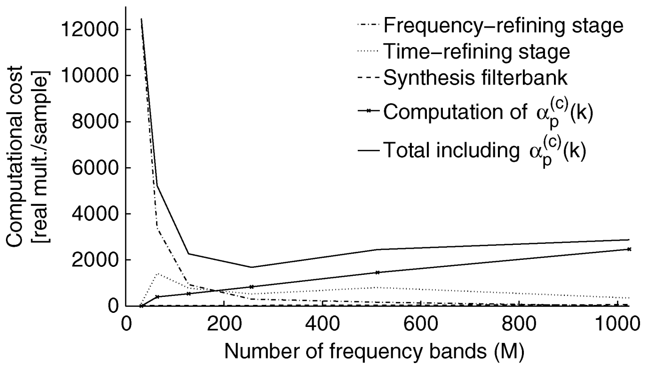

Fig. 6. Computational cost of the online synthesis algorithm versus the number of frequency bands M.

$l_m(z)$, shown in Fig. 3. In the case $M = 32$, the time-refining stage is not needed, since the time resolution is already fine enough without it. Hence its computation is not considered in the figures. Also, the use of the frequency-refining stage is not needed when $M = 1024$; hence, its computation is also not considered. Moreover, the removal of this stage implies that the input of the transfer matrices $\mathbf{T}^{(c)}$ is real-valued. Hence, the cost of their implementation is reduced by half.

In the case of offline synthesis, we see that the total complexity of the algorithm decreases as M increases, and this is mostly due to the complexity reduction of the frequency-refining stage. Hence, the optimal choice is $M = 1024$. This leads to a complexity of 396 multiplications per sample, which is split into 356 multiplications for the time-refining stages and 40 multiplications for the synthesis filterbank. For online synthesis, the optimal choice is $M = 256$, which results in an overall complexity of 1677 multiplications, split into 296 for the frequency-refining stages, 521 for the time-refining stages, 832 for the online computation of the synthesis coefficients $\alpha_p^{(c)}(k)$, and 28 for the synthesis filterbank.

Remark 6: Notice that the designs studied in Figs. 5 and 6 do not impose any constraint on the synthesis delay (latency). More precisely, as explained in Section III-B, the analysis filterbank $h_m(z)$ and the synthesis filterbank $l_m(z)$ form a perfect reconstruction pair. This implies that either the impulse response $h_m(t)$ or $l_m(t)$ or both have non-causal taps. This in turns implies that the matrix impulse responses $\mathbf{T}^{(c)}(t)$, defined in (17), also have a non-causal component. For the practical implementation, a delay needs to be added to make $\mathbf{T}^{(c)}(t)$ causal.[1] If δ denotes this delay, the synthesized signal $y(t)$ will have a delay of $\delta D = 3\delta M/4$ samples. Hence, if synthesis delay is a concern, this

[1]Strictly speaking, another delay needs to be added if $l_m(t)$ has non-causal taps. However, this is not required since the non-causality of $l_m(z)$ can be eliminated by adding a negative delay to $h_m(z)$.

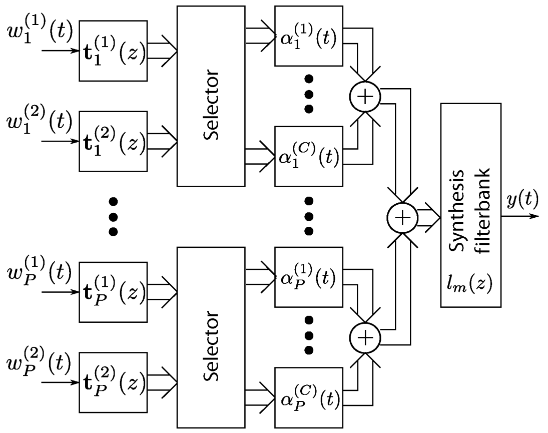

Fig. 7. Proposed offline synthesizer architecture.

may impose a constraint on the choice of M. In particular, for the optimal online design (i.e., with $M = 256$), this delay is 224 samples (~5 ms.), which is reasonable for most interactive applications. On the other hand, the optimal offline design (i.e., with $M = 1024$) has a latency of 2036 samples (~46 ms.), which might be too large. To reduce this latency, we can choose from Table I the offline design with $M = 512$ and $D = 367$. While this choice increases the complexity from 396 to 738 multiplications, it results in a latency of 734 samples (~17 ms), which meets the interactivity requirements.

D. Optimal Offline Synthesis Scheme

In view of Fig. 5, to minimize the complexity in the case of offline synthesis, we propose a design using $M = 1024$. Recall that in this case, the frequency-refining stage is removed. This is equivalent to choosing, in Fig. 3, $\mathbf{S}_p(z)$, for each $p = 1, \cdots, P$, as a column vector with its pth entry $[\mathbf{S}_p(z)]_p$ equal to one and zeros in its other entries (i.e., $[\mathbf{S}_p(z)]_p = 1$ and $[\mathbf{S}_p(z)]_i = 0$, for $i \neq p$). This implies that the shape of the spectral components $\phi_p(z)$ is directly determined by the synthesis filterbank $f_m(z)$, via $\phi_p(z) = |f_p(z)|^2$. Hence, to obtain the shapes shown in Fig. 4, when $M = 1024$, $f(z)$ is designed as a root raised cosine filter instead of a raised cosine filter. This results in the impulse response of $f(z)$ being much longer. We use an impulse response length of 16 384 samples. However, notice that these filters are only used to build the transfer matrices $\mathbf{T}^{(c)}$ in (17), and their influence in the overall complexity is only implicit in the computation of these matrices.

The resulting scheme is depicted in Fig. 7, where for each $p = 1, \cdots, P$, $w_p^{(1)}(t)$, and $w_p^{(2)}(t)$ denote scalar white random processes, and $\mathbf{t}_p^{(1)}$ and $\mathbf{t}_p^{(2)}$ denote the pth column of $\mathbf{T}^{(1)}$ and $\mathbf{T}^{(2)}$, respectively. Also, the block labeled *selector* constructs the output of the model $\mathbf{T}^{(c)}$, for all $c = 1, \cdots, C$, using only the outputs of $\mathbf{T}^{(1)}$ and $\mathbf{T}^{(2)}$, as explained in Remark 4.

E. Comparison With the Synthesis Method in [21]

The design proposed in [21] uses $M = 128$ and $D = 32$. While that design is different from the one considered in this correspondence (here we use $D = 108$ when $M = 128$), it is designed aiming at the same goal than the design with $M = 32$ in Fig. 5, namely, to avoid using a time-refining stage. Fig. 5 shows that this is not the optimal choice as far as complexity is concerned. In order to avoid having a very large complexity, only $P = 134$ spectral components were used in [21] (instead of $P = 1024$ as done in this work) and they were uniformly distributed in the linear scale at low frequencies and in the ERB scale at high frequencies. For comparison purposes, if we use the design proposed in [21], using $P = 1024$ spectral components

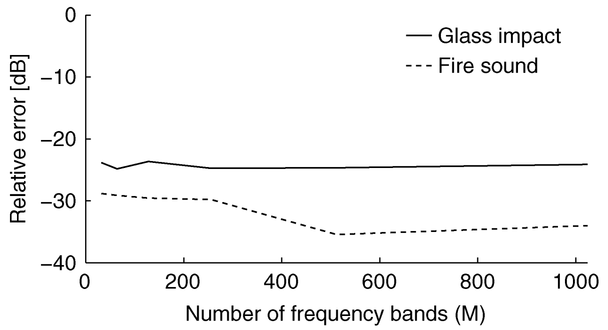

Fig. 8. Relative synthesis error versus the number of frequency bands M.

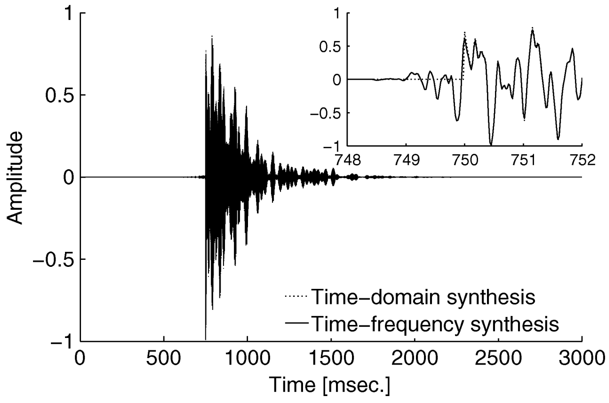

Fig. 9. Synthesis of a glass impact starting from 750 ms., and detail of the attack showing a settling time of about 1.5 ms. for the time–frequency method.

uniformly distributed in the linear scale, the resulting complexity would be 14 264 real multiplications per sample, i.e., comparable to that of the design with $M = 32$ shown in Fig. 5. This is about 8.5 times more complex than the proposed online design and 36 times more complex that the proposed offline design.

V. Numerical Experiments

A. Synthesis Examples

In order to illustrate the application of the proposed method, we synthesize two examples taken from [21]. The first is a blurry effect on a glass impact. This sound consists of narrow frequency bands at 1051, 1849, 3388, 5339, 7606, and 10 163 Hz, whose amplitude decay exponentially with time constants 4.948, 6.397, 10.78, 21.26, 47.49, and 110.4, respectively. We use the proposed offline synthesis scheme summarized in Section IV-D. To obtain a reference for comparison, we consider the *time domain* (brute force) method obtained by directly multiplying the outputs of the frequency-refining stages in Fig. 1, by the desired amplitude values, before addition. To do the comparison we average, over a 100 runs, the relative square error (difference) between the proposed synthesis method and the reference. Fig. 8 shows that this difference is about −23 dB for all values of M. Notice that, in view of the reference used, this error measures the performance lost due to the time resolution limitation of the proposed method. Hence, it is mostly due to the error at the impact instant.

For $M = 1024$, the synthesized signals and their spectra are shown in Figs. 9 and 10, respectively. For this particular run, the relative square error is −23.36 dB. Fig. 9 shows a detail of the impact instant. We see that the time domain method reacts instantaneously to the impact, while the proposed time–frequency method reacts in about 1.5 ms. Fig. 10 shows that the spectra of both methods are similar in shape, except for some valleys which appear between contiguous peaks in the time–frequency method. These valleys do not produce

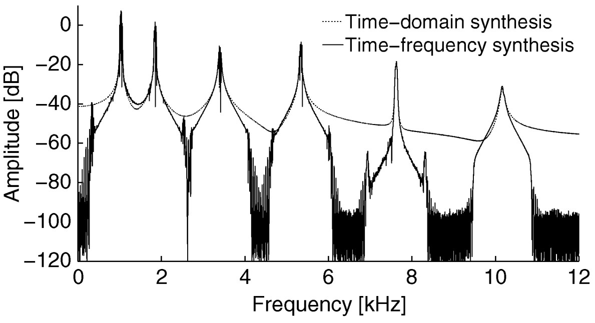

Fig. 10. Spectra of the synthesized glass impact sounds.

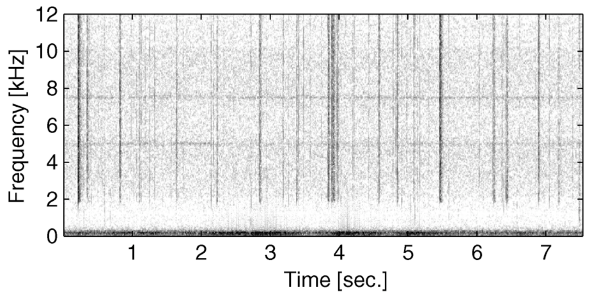

Fig. 11. Spectrogram of fire sound synthesized using the time domain method.

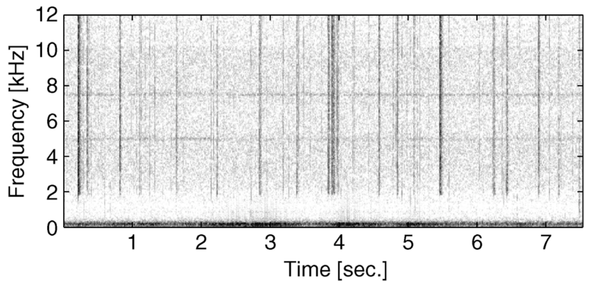

Fig. 12. Spectrogram of fire sound synthesized using the proposed time–frequency domain method.

any perceptual artifact, and result from the truncation of (16) to $\bar{C}$ terms, as described above.

The second example is the synthesis of fire sound. This sound is formed by three components. The first is the combustion, which is a narrow component at very low frequency. The second is the hissing, which is formed by two very narrow components at relatively high frequency (around 5 and 7.5 kHz). The third components is formed by the cracklings, which present very short transients. The relative square error between the synthesis using the time domain (brute force) method and the proposed offline time–frequency method, is shown in Fig. 8. We see that this error is between -30 dB and -35 dB for all values of M. For $M = 1024$, the spectrograms for the time domain and the proposed methods are shown in Figs. 11 and 12, respectively. We see that the spectrogram obtained with the proposed method clearly shows the presence of all three components, and that it reasonably resembles the reference spectrogram obtained using the time domain method.

The sound synthesis examples presented above, as well as other examples including waves, wind, whoosh, stones, etc., can be found at http://www.lma.cnrs-mrs.fr/~kronland/UnifiedTFSynth/sounds.html.

B. Comparison With the Frequency-Refining Technique Proposed in [21]

As mentioned in Section III-A, a frequency-refining technique was proposed in [21], which consists in the concatenation of an analysis filterbank, a transfer matrix and a synthesis filterbank. In this section,

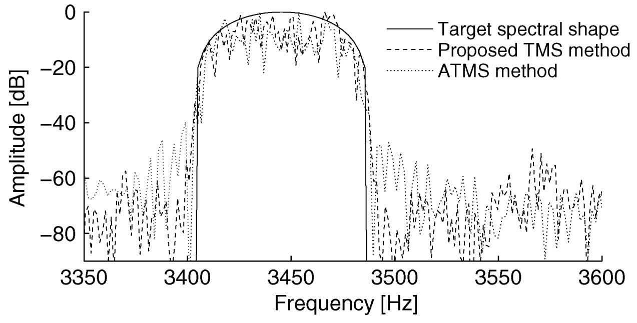

Fig. 13. Comparison of the ATMS and TMS frequency-refining techniques.

we denote this technique by ATMS. In this correspondence we propose an alternative technique which does not require the analysis filterbank stage. We denote the proposed technique by TMS. The advantage of the TMS technique is that it permits applying a real-valued signal at the transfer matrix input, which reduces by two its processing complexity. In this section, we compare the performance and complexity of these two approaches.

As target spectral shape we consider the 11th band of the design described in Section IV-C, i.e., $\phi_{11}(z)$, which has center frequency 3445 Hz, and bandwidth 86 Hz. For both methods we use the configuration in [21], i.e., we use $M = 128$, $D = 32$ and the filterbanks are designed using digital prolate spheroidal sequences of order 0 [35] and length 128 samples. Both methods are designed using an iterative procedure, which in the case of the TMS method is described by (11)–(13). In both cases, we stop these iterations when the relative energy difference between $\phi_{11}(z)$ and the produced spectrum is smaller than -40 dB. The spectral shapes obtained using both methods, together with the target shape, are shown in Fig. 13. We see that both shapes reasonably resemble each other. However, the computation of the ATMS method requires 60 real multiplications per sample for the analysis filterbank, 28 multiplications for the transfer matrix and 60 multiplications for the synthesis filterbank, while that of the TMS method requires only 14 multiplications for the transfer matrix and 60 multiplications for the synthesis filterbank. In a scheme like the one described in Section IV-C, the analysis and synthesis filterbanks are only applied once for all spectral components. Hence, not using analysis filterbank is not a major advantage in this context. However, the TMS method has still the advantage of reducing by two the complexity of the transfer matrix.

VI. Conclusion

We proposed a noisy sound synthesis method, which carries out the synthesis task in the time–frequency domain, and which is able to achieve any time and frequency resolution, for any number of frequencies used for time–frequency processing. To do so, we introduced time- and frequency-refining techniques to overcome the inherent time and frequency resolutions limitations resulting from time–frequency processing. Using this property, we studied the number of frequencies which minimizes the overall complexity of the synthesis algorithm. We did so for the two scenarios resulting from whether the synthesis parameters are computed offline or online. We compare the complexity of the proposed designs with that of a design previously proposed by the authors. That design was done using only frequency-refining, and therefore a small number of frequencies was used to achieve the desired time resolution. The comparison shows that the newly proposed method leads to a major complexity reduction.

Appendix

Proof of (14): Let $y_m(k)$ denote the mth component of the output of the filterbank $h_m(z)$ in Fig. 2. Let $x_n(k)$ denote the nth component of the input of the filterbank $f_m(z)$, and let $x(t)$ denote its output. We have that

$$\begin{aligned} y_m(k) &= (h_m * y)(kD) \\ &= \sum_{t=-\infty}^{\infty} h_m(kD-t)y(t) \\ &= \sum_{t=-\infty}^{\infty} h_m(kD-t)a(t)x(t) \\ &= \sum_{t=-\infty}^{\infty} h_m(kD-t)a^{(k)}(kD-t)x(t) \\ &= \left(\left(a^{(k)}h_m\right) * x\right)(kD) \\ &= \left(\left(a^{(k)}h_m\right) * \sum_{n=1}^{M} f_n * \uparrow_D \{x_n\}\right)(kD). \end{aligned}$$

Let $\xi_{k.m.n} = (a^{(k)}h_m) * f_n$. Then,

$$\begin{aligned} y_m(k) &= \left(\sum_{n=1}^{M} \xi_{k.m.n} * \uparrow_D \ \{x_n\}\right)(kD) \\ &= \sum_{n=1}^{M}\sum_{t=-\infty}^{\infty} \uparrow_D \ \{x_n\}(t)\xi_{k.m.n}(kD-t) \\ &= \sum_{n=1}^{M}\sum_{l=-\infty}^{\infty} x_n(l)\xi_{k.m.n}((k-l)D) \\ &= \sum_{n=1}^{M} \left(\downarrow_D \ \{\xi_{k.m.n}\} * x_n\right)(k) \end{aligned}$$

where $\downarrow_D \ \{\cdot\}$ denotes the downsampling operation with factor D. Hence, $[\mathbf{T}]_{m,n}(l,k) = \downarrow_D \ \{\xi_{k.m.n}\}(l)$, and (14) follows. ■

Proof of (16): We have that

$$\begin{aligned} [\mathbf{T}]_{m,n}(l,k) &= \left(\left(a^{(k)}h_m\right) * f_n\right)(lD) \\ &= \sum_{t=-\infty}^{\infty}\sum_{c=1}^{C} \alpha^{(c)}(k)e^{j2\pi\frac{c-1}{C}t}h_m(t)f_n(lD-t) \\ &= \sum_{c=1}^{C} \alpha^{(c)}(k)\left(h_m^{(c)} * f_n\right)(lD) \\ &= \sum_{c=1}^{C} \alpha^{(c)}(k)\left[\mathbf{T}^{(c)}\right]_{m,n}(l). \end{aligned}$$

■

References

[1] K. van den Doel, "Physically based models for liquid sounds," in *Proc. ICAD 04-10th Meeting Int. Conf. Auditory Display*, 2004.

[2] C. Zheng and D. L. James, "Harmonic fluids," *ACM Trans. Graphics (Proc. SIGGRAPH'09)*, vol. 28, no. 3, pp. 37:1–37:12, 2009.

[3] W. Moss, H. Yeh, J.-M. Hong, M. C. L. Ming, and D. Manocha, "Sounding liquids: Automatic sound synthesis from fluid simulation," *ACM Trans. Graph.*, vol. 29, pp. 21:1–21:13, Jul. 2010.

[4] N. Bonneel, G. Drettakis, N. Tsingos, I. Viaud-Delmon, and D. James, "Fast modal sounds with scalable frequency-domain synthesis," *ACM Trans. Graphics (Proc. SIGGRAPH '08)*, vol. 27, no. 3, 2008.

[5] C. Zheng and D. L. James, "Rigid-body fracture sound with precomputed soundbanks," *ACM Trans. Graphics (Proc. SIGGRAPH '10)*, vol. 29, no. 3, pp. 69:1–69:13, 2010.

[6] C. Zheng and D. L. James, "Toward high-quality modal contact sound," *ACM Trans. Graphics (Proc. SIGGRAPH '11)*, vol. 30, no. 4, 2011.

[7] Y. Dobashi, T. Yamamoto, and T. Nishita, "Real-time rendering of aerodynamic sound using sound textures based on computational fluid dynamics," *ACM Trans. Graphics (Proc. SIGGRAPH '03)*, vol. 22, no. 3, pp. 732–740, 2003.

[8] A. Farnell, *Designing sound*. Cambridge, MA: MIT Press, 2010.

[9] J. Chadwick and D. L. James, "Animating fire with sound," *ACM Trans. Graphics (Proc. SIGGRAPH 2011)*, vol. 30, no. 4, 2011.

[10] D. Rocchesso and F. Fontana, "The Sounding Object," [Online]. Available: http://www.soundobject.org/ 2003

[11] C. Roads, *The Computer Music Tutorial*, 5th ed. Cambridge, MA: MIT Press, 2000.

[12] P. R. Cook, "Physically informed sonic modeling (phism): Synthesis of percussive sounds," *Comput. Music J.*, vol. 21, no. 3, pp. 38–49, 1997.

[13] C. Verron, M. Aramaki, R. Kronland-Martinet, and G. Pallone, "Controlling a spatialized environmental sound synthesizer," in *Proc. IEEE Workshop Applicat. Signal Process. Audio Acoust.*, 2009, pp. 321–324.

[14] X. Serra and J. O. Smith, "Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition," *Comp. Music. J.*, vol. 14, no. 4, pp. 12–24, 1990.

[15] A. Misra, P. R. Cook, and G. Wang, "A new paradigm for sound design," in *Proc. Int. Conf. Digital Audio Effects (DAFx06)*, 2006.

[16] C. Verron, M. Aramaki, R. Kronland-Martinet, and G. Pallone, "A 3D Immersive synthesizer for environmental sounds," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 18, no. 6, pp. 1550–1561, Aug. 2010.

[17] D. Schwarz and M. Wright, "Extensions and applications of the SDIF sound description interchange format," in *Proc. ICMC*, 2000.

[18] X. Rodet and P. Depalle, "Spectral envelopes and inverse FFT synthesis," in *Proc. 93rd AES Conv.*, 1992.

[19] X. Amatriain, J. Bonada, A. Loscos, and X. Serra, *DAFX: Digital Audio Effects*. New York: Wiley, 2002, ch. Spectral Processing.

[20] K. van den Doel, P. G. Kry, and D. K. Pai, "Foleyautomatic: Physically based sound effects for interactive simulation and animation," in *Proc. 28th Annu. Conf. Comput. Graphics Interactive Tech.*, 2001, pp. 537–544.

[21] D. Marelli, M. Aramaki, R. Kronland-Martinet, and C. Verron, "Time–frequency synthesis of noisy sounds with narrow spectral components," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 18, no. 8, pp. 1929–1940, Nov. 2010.

[22] T. F. Quatieri, *Discrete-Time Speech Signal Processing: Principles and Practice*, 1st ed. Upper Saddle River, NJ: Prentice-Hall, 2001, vol. 11.

[23] P. Vaidyanathan, *Multirate Systems and Filterbanks*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[24] A. Sayed and T. Kailath, "A survey of spectral factorization methods," *Numer. Linear Algebra With Applicat.*, vol. 8, no. 6-7, pp. 467–496, 2001.

[25] A. Ben-Israel and T. N. E. Greville, *Generalized Inverses*, ser. ser. CMS Books in Mathematics/Ouvrages de Mathématiques de la SMC, 2nd ed. New York: Springer-Verlag, 2003, vol. 15, ch. Theory and Applications.

[26] K. Gröchenig, *Foundations of Time–Frequency Analysis*, ser. Applied and Numerical Harmonic Analysis. Boston, MA: Birkhäuser, 2001.

[27] J. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2231–2242, Oct. 2004.

[28] G. Davis, S. Mallat, and Z. Zhang, "Adaptive time–frequency decompositions," *Opt. Eng.*, vol. 33, p. 2183, 1994.

[29] Y. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation withapplications to wavelet decomposition," in *Conf. Rec. 27th Asilomar Conf. Signals, Syst., Comput.*, 1993, pp. 40–44.

[30] J. Proakis, *Digital Communications*, 4th ed. New York: McGraw-Hill, 2000, vol. 8.

[31] M. Vetterli and J. Kovacevic, *Wavelets and Subband Coding (Prentice Hall Signal Processing Series)*. Englewood Cliffs, NJ: Prentice-Hall, 1995, vol. 4.

[32] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing (2nd Edition) (Prentice-Hall Signal Processing Series)*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 1999, vol. 1.

[33] S. Weiss and R. Stewart, "Fast implementation of oversampled modulated filter banks," *Electron. Lett.*, vol. 36, no. 17, pp. 1502–1503, Aug. 2000.

[34] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 3rd ed. Englewood-Cliffs, NJ: Prentice-Hall, 1996.

[35] D. Slepian, "Prolate spheroidal wave functions, Fourier analysis, and uncertainty. V-The discrete case," *Bell Syst. Tech. J.*, vol. 57, pp. 1371–1430, 1978.