Journal of New Music Research
2007, Vol. 36, No. 4, pp. 241 – 250

Routledge
Taylor & Francis Group

Musical Tapestry: Re-composing Natural Sounds†

Ananya Misra[1], Ge Wang[2] and Perry Cook[1]

[1]Princeton University, USA, [2]Stanford University, USA

Abstract

A system to aid composition with analysis, transformation, and re-synthesis of natural sounds is described. Sinusoidal analysis is used to isolate and extract deterministic sounds, and transients are also isolated/extracted, leaving the stochastic background which is parameterized by wavelet tree analysis. All of these components become templates for the synthesis phase, which is controlled (1) by placing templates on timelines or in groups, (2) by real-time manipulation of parameters, and (3) via scripting using the ChucK language. The result is a flexible ''workbench'' for doing modern day musique concrète or acousmatic composition, sound design, and other sonic sculpting tasks.

1. Motivation

Around 1950, Pierre Schaeffer developed musique concrète (Schaeffer, 1950, 1952). Unlike traditional music, musique concrète starts with existing or concrete recorded sounds, which are organized into abstract musical structures. The existing recordings often include natural and industrial sounds that are not conventionally musical, but can be manipulated to make music, either by editing magnetic tape or now more commonly through digital sampling. Typical manipulations include cutting, copying, reversing, looping and changing the speed of recorded segments. Today, several other forms of electronic/electroacoustic music also involve manipulating a set of recorded sounds. Acousmatic music (Dhomont, 1995), for instance, evolved from musique concrète and refers to compositions designed for environments that emphasize the sound itself rather than the performance-oriented aspects of the piece.

The acoustic ecology (Schafer, 1977) movement gave rise to soundscape composition (Truax, 2002) or the creation of realistic soundscapes from recorded environmental audio. One of the key features of soundscape composition, according to Truax, is that ''most pieces can be placed on a continuum between what might be called 'found sound' and 'abstracted' approaches'' (Truax, 2002, p. 6). However, while ''contemporary signal processing techniques can easily render such sounds unrecognizable and completely abstract'' (Truax, 2002, p. 6) a soundscape composition piece is expected to remain recognizable even at the abstract end of the continuum.

Sound designers for movies, theatre and art often have a related goal of starting with real world sounds and creating emotionally evocative sound scenes, which are still real, yet transformed and transformative. Classic examples include layering of various sounds to build up the final wave sound in *The Perfect Storm* (''Voice of the Beast'', 2000), and incorporating a helicopter theme into the sound design for *Black Hawk Down* (Rudy, 2004). These sound designers are ''sound sculptors'' as well, but transform sounds to enhance or create a sense of reality, rather than for purely musical purposes.

Artists from all of the above backgrounds share the process of manipulating recordings, but aim to achieve different effects. We present a single framework for starting with recordings and producing sounds that can

†Winner of the Journal of New Music Research Distinguished Paper Award at the *International Computer Music Conference* 2006.

Correspondence: Ananya Misra, Department of Computer Science, Princeton University, 35 Olden Street, Princeton, NJ 08540, USA. E-mail: amisra@cs.princeton.edu
Ge Wang is currently an assistant professor at the Center for Computer Research in Music and Acoustics (CCRMA), Stanford University, USA. E-mail: ge@ccrma.stanford.edu
Perry Cook is also in Department of Music, Princeton University, USA. E-mail: prc@cs.princeton.edu

DOI: 10.1080/09298210801984839

lie anywhere on a "found" to "unrecognizable" continuum. "Found" sounds can be modified in subtle ways or extended indefinitely, while moving towards the "unrecognizable" end of the spectrum unleashes a range of manipulations beyond time-domain techniques. In fact, the same set of techniques applies throughout the continuum, differing only in how each is used. We call this framework TAPESTREA: Techniques and Paradigms for Expressive Synthesis, Transformation and Rendering of Environmental Audio.

The TAPESTREA system integrates sinusoidal analysis, stochastic background modelling, transient detection, and a new class of user interface that lends itself to any composition that originates in recorded environmental audio. This envelops a novel form of musique concrète that extends to manipulations in the frequency as well as time domain. Advantages of the TAPESTREA approach include:

- TAPESTREA lets the sound sculptor select a region in both time and frequency, essentially specifying, "Give me *this* part of *that* sound", to extract a reusable *sound template*.
- TAPESTREA defines three fundamental types of sound components/templates, based on the modelling techniques for which they are best suited. *Sinusoidal, transient*, and *stochastic background* components are modelled separately, using methods to which they are most amenable, leading to specialized control and more powerful transformations on each type.
- To realize these ideas, TAPESTREA provides a set of interfaces that allow the sound designer or composer to assert parametric control over each phase in the process, from component extraction to the final resynthesis.

TAPESTREA manipulates sounds in several phases (Figure 1). In the analysis phase, the sound is separated into reusable components that correspond to individual foreground events or background textures. In the synthesis phase, these components are transformed, combined and re-synthesized using time- and frequency-domain techniques that can be controlled on multiple levels. While we highlight the synthesis methods here, the analysis phase is also integral as it enables the most flexible means for dealing with real-world sonic material.

2. Related work

Related audio processing techniques include sinusoidal modelling (McAulay & Quatieri, 1986), where a sound is decomposed into a set of sine waves at varying frequencies. Spectral modelling synthesis (Serra, 1989) extends sinusoidal modelling by separating a sound into sinusoids (or the deterministic component) plus noise, where shaped noise models parts of the sound that are not characterized well by sine waves. This technique was originally used for modelling instrument sounds.

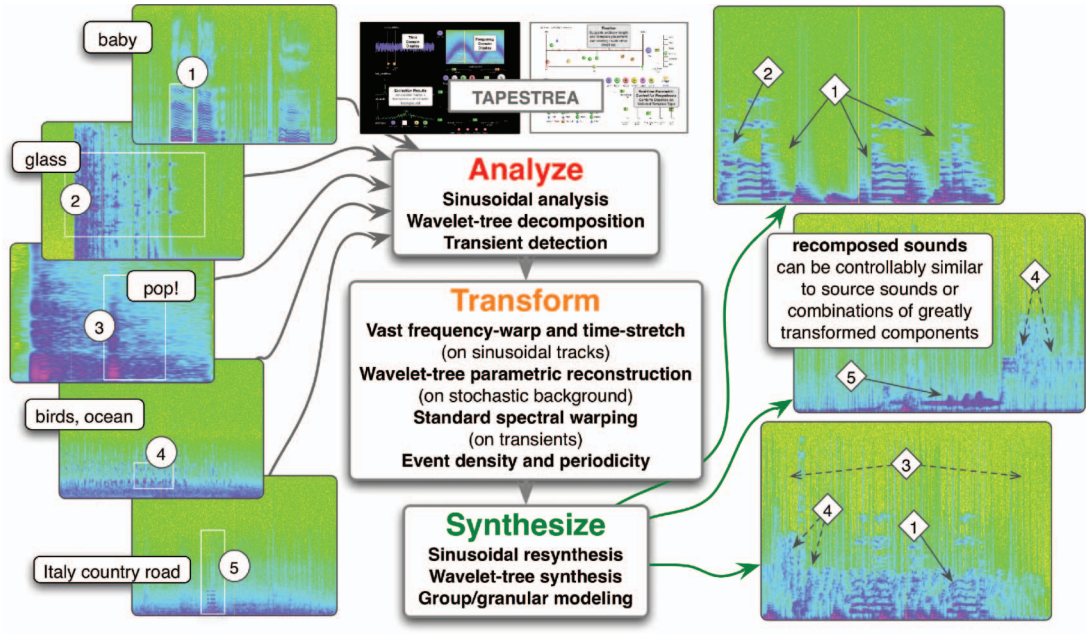

Fig. 1. Creating musical tapestries. User-selected regions of input sounds (left) are analysed into reusable templates, which are separately transformed and re-synthesized into new sounds (right). Numbered diamonds (right) correspond to instances of original sound components (circle, left). The framework allows flexible control at every stage in the process.

Past work on analysing transients, or short spurts of noise, include detection and modelling techniques that can sometimes be understood as a time-domain counterpart of sinusoidal modelling (Verma & Meng, 1998). Methods for onset detection in music (Bello et al., 2005) are also closely related.

Also relevant is granular synthesis (Truax, 1990; Roads, 2002). This synthesis technique functions in the time domain and involves continuously controlling very brief sonic events, or sound grains. The temporal positioning and audio properties of the sound grains determine the characteristics of the synthesized sound.

TAPESTREA employs aspects of all of the above, using the separation techniques on environmental sounds and controlling the temporal placement of resulting events. Another technique used is an extension of a wavelet tree learning algorithm (Dubnov et al., 2002) for sound texture synthesis. This method performs wavelet decomposition on a sound clip and uses machine learning on the wavelet coefficients to generate similar non-repeating sound texture. The algorithm works well for sounds that are mostly stochastic, but can break extended pitched portions in objectionable ways. It can also be slow in its original form. TAPESTREA takes advantage of this technique by improving the speed of the algorithm, and only using it on the types of (non-deterministic) sound for which it works well.

There exist other tools for spectral analysis, transformation, and re-synthesis, such as AudioSculpt (Bogaards et al., 2004), SPEAR (Klingbeil, 2005), and the CLAM library (Amatriain & Arumi, 2005). However, these generally have an automated analysis phase and thus do not offer the same level of flexible, interactive control over all stages of audio processing. They also lack a framework for handling transients and stochastic background components.

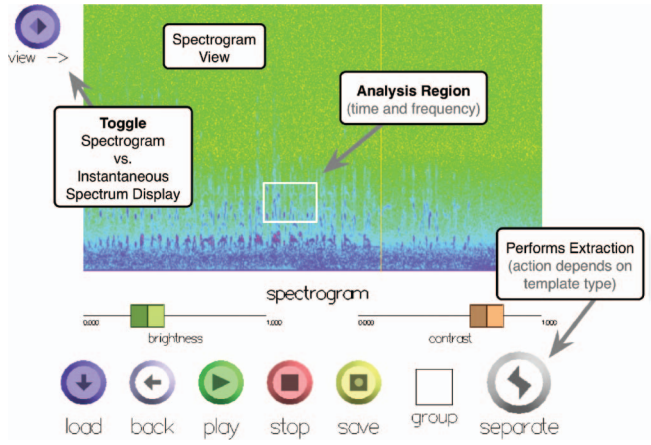

Fig. 2. Spectrogram view in analysis face.

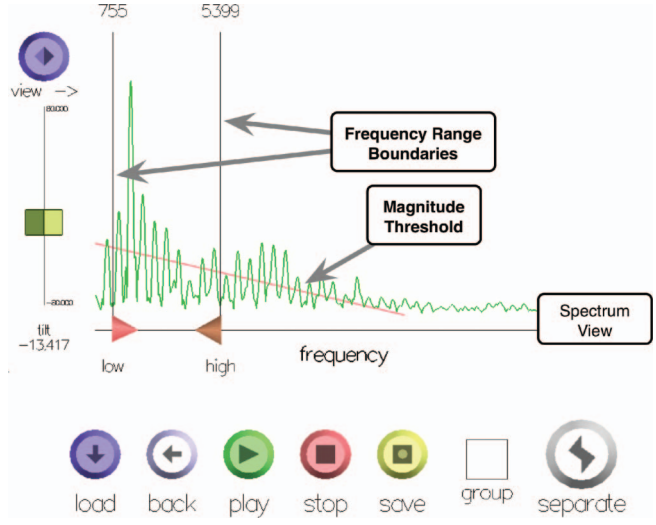

Fig. 3. Spectrum view in analysis face. The slanting line specifies the magnitude threshold.

3. Analysis phase

TAPESTREA starts by separating a recording into *sinusoidal events* or stable sinusoidal components of the sound, *transient events* or brief noisy bursts of energy, and the remaining *stochastic background* or din. This separation can be parametrically controlled and takes place in the analysis phase. In a sense, boundaries between component types are not rigid, but are interactively defined by the user.

The analysis interface is shown in the accompanying figures. A loaded sound is simultaneously displayed as a waveform and a spectrogram (Figure 2). The spectrogram display can also be toggled with a frame-by-frame spectrum view (Figure 3). Selecting a rectangle on the spectrogram, or selecting an analysis region on the waveform and the frame-by-frame spectrum, limits the analysis to the associated time and frequency ranges, facilitating the selection and extraction of specific events.

Sinusoidal events are foreground events extracted by sinusoidal modelling based on the spectral modelling framework (Serra, 1989). Overlapping frames of the sound are transformed into the frequency domain using the FFT. For each spectral frame, the n highest peaks above a specified magnitude threshold (Figure 3) are recorded, where n can range from 1 to 50. These peaks can also be loaded from a preprocessed file. The highest peaks from every frame are then matched across frames by frequency, subject to a controllable "frequency sensitivity" threshold, to form sinusoidal tracks. Tracks can be "mute" (below the magnitude threshold) for a specified maximum number of frames, or can be discarded if they fail to satisfy a minimum track length requirement (Figure 4). Un-discarded tracks are optionally grouped (Ellis, 1994; Melih & Gonzalez, 2000) by

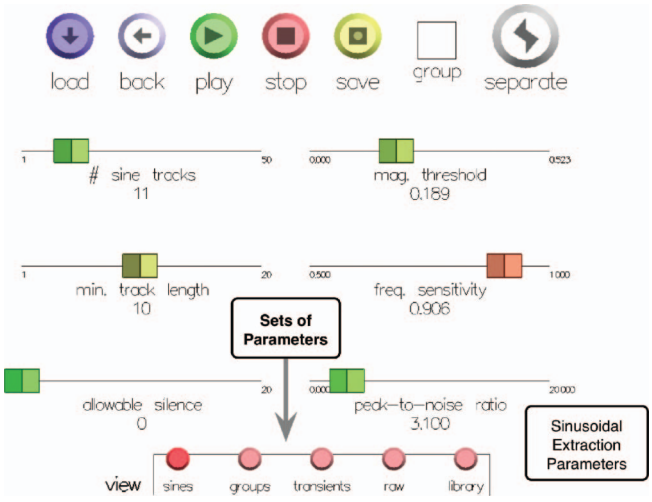

Fig. 4. Sliders for sinusoidal analysis.

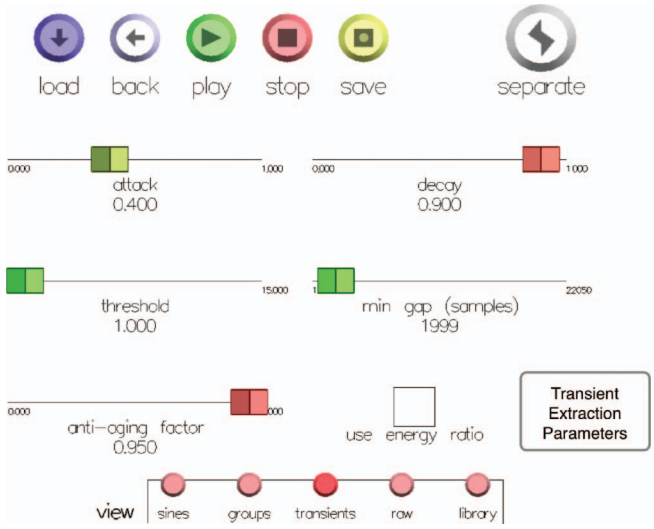

Fig. 5. Transient analysis sliders.

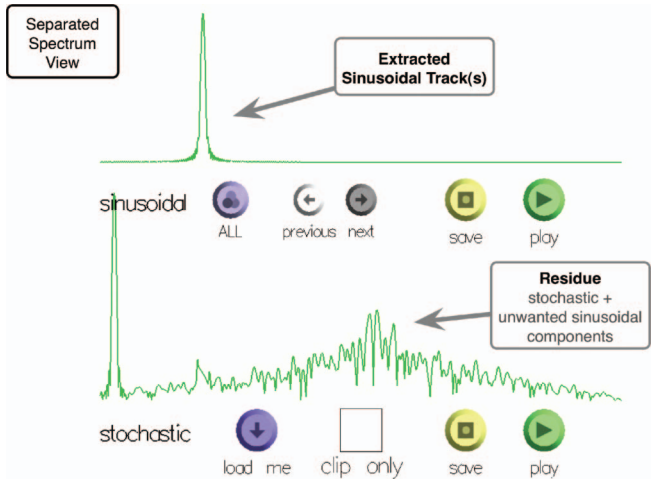

Fig. 6. Spectrum of separated sinusoidal peaks (top) and stochastic residue (bottom).

harmonicity, common amplitude and frequency modulation, and common onset/offset, to form sinusoidal events, which are essentially collections of related sinusoidal tracks. If the grouping option is not selected, all the tracks found are together interpreted as a single sinusoidal event. After the separation, the sinusoidal tracks found are marked on the spectrogram display. Each sinusoidal event can be individually played and saved as a template for use in the synthesis phase.

Transient events or brief noisy foreground events can be detected in the time-domain by observing changes in signal energy over time (Verma & Meng, 1998; Bello et al., 2005). TAPESTREA analyses the recorded sound using a non-linear one-pole envelope follower filter with a sharp attack and slow decay and finds points where the derivative of the envelope is above a threshold. These points mark sudden increases in energy and are interpreted as transient onsets. A transient event is considered to last for up to half a second from its onset. The exact transient length, as well as the threshold, and filter parameters can all be modified in real-time via sliders (Figure 5). Detected transients can be individually replayed and saved as templates.

The *stochastic background* represents parts of the recording that constitute background noise, and is obtained by removing the detected sinusoidal and transient events from the initial sound. Sinusoidal events are removed by eliminating the peaks of each sinusoidal track from the corresponding spectral frames; the magnitudes of the bins beneath the peak are smoothed down, while the phases in these bins are randomized (Figure 6). Transient events, in turn, are removed in the time-domain by applying wavelet tree learning (Dubnov, 2002) to generate a sound clip that resembles nearby transient-free segments of the recording. This synthesized ''clean'' background replaces the samples containing the transient event to be removed. Once separated, the stochastic background can be saved, played, or loaded into the interface for further iterative analysis.

Separating a sound into components in this way has several advantages. The distinction between foreground and background components is semantically clear to humans, who can therefore work within the framework with a concrete understanding of what each component represents. The different component types are also stored and processed separately according to their defining characteristics, thus allowing flexible transformations on individual components. Each transformed component can be saved as a template and later reloaded, reused, copied, further transformed, or otherwise treated as a single object. In addition, the act of separating a sound into smaller sounds makes it possible to ''re-compose'' them into a variety of pieces by combining templates in diverse ways.

4. Synthesis phase

Once the components of a sound have been separated and saved as templates, TAPESTREA allows each template to be transformed and synthesized individually. The synthesis interface (Figure 7) provides access to the current library of saved templates, displayed as objects (Figure 8). Templates saved to file from prior sittings can be loaded into the library, too. Selecting any template in the library displays a set of transformation and synthesis parameters suited to the template type. A selected template can be synthesized to generate sound at any time, including while its transformation parameters are being modified. At this point, TAPESTREA also offers additional synthesis templates to control the placement or distribution of basic components in a composition. Thus, components can be manipulated individually and in groups, modelling both single sound and group characteristics. The transformation and synthesis options for the different template types are as follows.

4.1 Sinusoidal events

Sinusoidal events are synthesized from their tracks via sinusoidal re-synthesis. Frequency and magnitude between consecutive frames in a track are linearly interpolated, and time-domain samples are computed from this information.

The track representation allows considerable flexibility in applying frequency and time transformations on a sinusoidal event. The event's frequency can be linearly scaled before computing the time-domain samples, by multiplying the frequency at each point on its tracks by a specified factor. Similarly, the event can be stretched or shrunk in time by scaling the time values in the time-to-frequency trajectories of its tracks. This works for almost any frequency or time scaling factor without producing artifacts. Frequency and time transformations can take place in real-time in TAPESTREA, allowing an event to be greatly stretched, shrunk or pitch shifted even as it is being synthesized.

4.2 Transient events

Since transient events are brief by definition, TAPESTREA stores them directly as time-domain audio frames. Synthesizing a transient event without any transformations, therefore, involves playing back the samples in the audio frame.

In addition, TAPESTREA allows time-stretching and pitch-shifting in transient events as well. This is implemented using a phase vocoder (Dolson, 1986). While a phase vocoder itself does not impose a limit on the scaling range, it is more computationally expensive than the transformations on sinusoidal events, and the results may often sound less clean. This is because the sinusoidal tracks drive a sine oscillator bank, allowing smooth frequency and amplitude transitions with no need to store phase information, whereas a phase vocoder would require an extremely small hop size to achieve a similar effect. Hence, to facilitate fast interactive transformations on transients, TAPESTREA

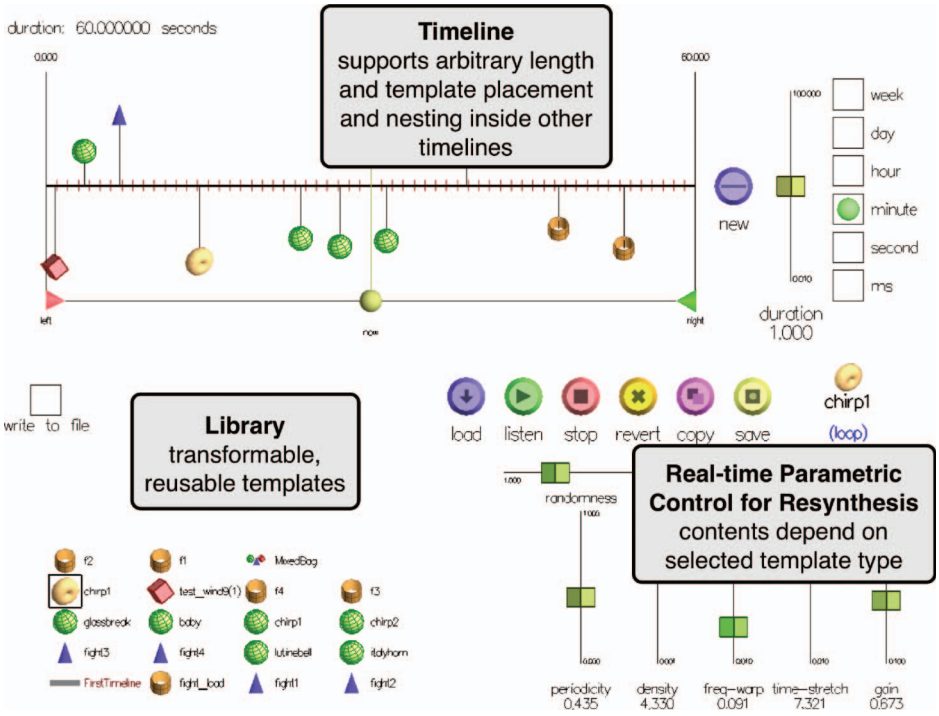

Fig. 7. Screenshot of transformation + synthesis interface.

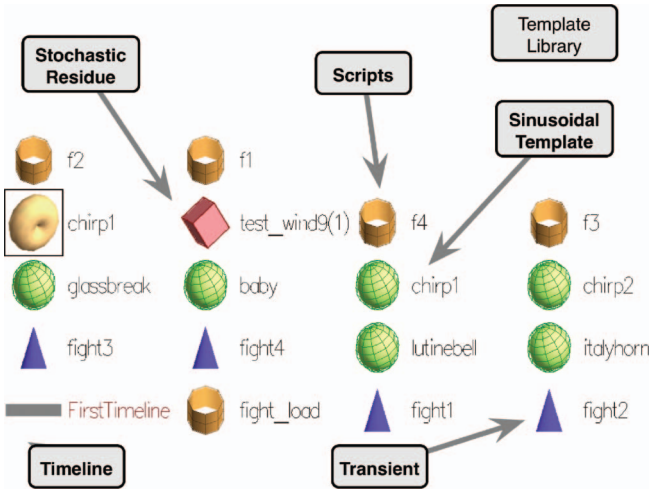

Fig. 8. Library of saved templates.

limits their scaling factors to a range smaller than what is available for sinusoidal events, yet large enough to create noticeable effects.

Transient events by nature can also act as "grains" for traditional granular synthesis (Truax, 1990; Roads, 2002). The transformation tools for transients, along with the additional synthesis templates described in Sections 4.4 to 4.6, can thus provide an interactive "granular synthesis" interface.

4.3 Stochastic background

The internal representation of a *stochastic background* template begins with a link to a sound file containing the related background component extracted in the analysis phase. However, merely looping through this sound file or randomly mixing segments of it does not produce a satisfactory background sound. Instead, our goal here is to generate ongoing background that sounds controllably similar to the original extracted stochastic background.

Therefore, the stochastic background is synthesized from the saved sound file using an extension of the wavelet tree learning algorithm (Dubnov et al., 2002). In the original algorithm, the saved background is decomposed into a wavelet tree where each node represents a coefficient, with depth corresponding to resolution. The wavelet coefficients are computed using the Daubechies wavelet with 5 vanishing moments. A new wavelet tree is then constructed, with each node selected from among the nodes in the original tree according to similarity of context. A node's context includes its chain of ancestors as well as its first k predecessors – nodes at the same level as itself but preceding it in time (Figure 9). The context of the next node for the new tree is compared to the contexts of nodes in the original tree, yielding a distance value for each original tree node considered. Eventually, the next new tree node is selected from among those original tree nodes whose distance values fall below a

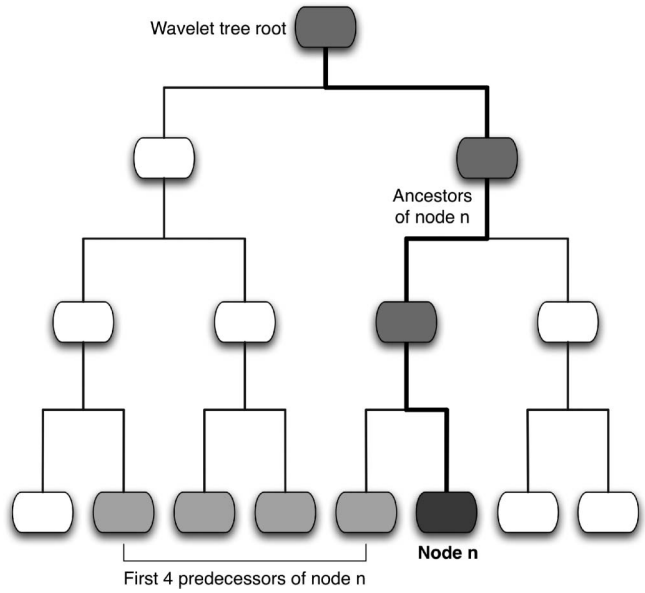

Fig. 9. Context of a given node n (coloured dark grey) in a wavelet tree. The ancestors of n are the nodes encountered in the path (marked with thick lines and medium grey colouring) between n and the root of the wavelet tree. The predecessors (coloured light grey) are nodes at the same level as n but preceding it in time.

specified threshold. The learning algorithm also takes into account the amount of randomness desired. Finally, the new wavelet tree undergoes an inverse wavelet transform to provide the synthesized time-domain samples. This learning technique works best with the separated stochastic background as input, where the sinusoidal events it would otherwise chop up have been removed.

TAPESTREA uses a modified and optimized version of the algorithm, which follows the same basic steps but varies in details. For instance, the modified algorithm includes the option of incorporating randomness into the first level of learning, and also considers k as dependent on node depth rather than being constant. More importantly, it optionally avoids learning the coefficients at the highest resolutions. These resolutions roughly correspond to high frequencies, and randomness at these levels does not significantly alter the results, while the learning involved takes the most time. Optionally stopping the learning at a lower level thus optimizes the algorithm and allows it to run in real-time.

Further, TAPESTREA offers interactive control over the learning parameters in the form of "randomness" and "similarity" parameters. The size of a sound segment to be analysed as one unit can also be controlled, and results in a "smooth" synthesized background for larger sizes versus a more "chunky" background for smaller sizes. Creatively manipulating these parameters can, in fact, yield interesting musical compositions generated through "stochastic background" alone.

4.4 Event loops

Event loops (Figure 10) are synthesis templates designed to facilitate the parametric repetition of a single event. Any sinusoidal or transient event template can be formed into a loop. When the loop is played, instances of the associated event are synthesized at the specified density and periodicity, and within a specified range of random transformations. These parameters can be modified while the loop is playing, to let the synthesized sound change gradually.

The density refers to how many times the event is repeated per second, and could be on the order of 0.001 to 1000. At the higher densities, and especially for transient events, the synthesized sound is often perceived as continuous, thus resembling granular synthesis.

The periodicity, ranging from 0 to 1, denotes how periodic the repetition is, with a periodicity of 1 meaning that the event is repeated at fixed time intervals. The interval between consecutive occurrences of an event is generally determined by feeding the desired periodicity and density into a Gaussian random number generator. It is straightforward to replace this generator with one that follows a Poisson or other user-specified probability distribution.

In addition to the parameters for specifying the temporal placement of events, TAPESTREA allows each instance of the recurring event to be randomly transformed within a range. The range is determined by selected average frequency- and time-scale factors, and a randomness factor that dictates how far an individual transformation may vary from the average. Individual transformation parameters are uniformly selected from within this range. Apart from frequency and time scaling, the gain and pan of event instances can also randomly vary in the same way.

4.5 Timelines

While a loop parametrically controls the repetition of a single event, with some amount of randomization, a timeline allows a template to be explicitly placed in time, in relation to other templates. Any number of existing templates can be added to a timeline, as well as deleted from it or re-positioned within it once they have been added.

A template's location on the timeline indicates its onset time with respect to when the timeline starts playing. When a timeline is played, each template on it is synthesized at the appropriate onset time, and is played for its duration or until the end of the timeline is reached. The duration of the entire timeline can be on the order of milliseconds to weeks, and may be modified after the timeline's creation.

TAPESTREA also allows the placement of timelines within timelines (or even within themselves). This allows for template placement to be controlled at multiple timescales or levels, making for a "multiresolution synthesis".

4.6 Mixed bags

Another template for synthesis purposes is the mixed bag (Figure 11), which is designed to control the relative densities of multiple, possibly repeating, templates. Like a timeline, a mixed bag can contain any number of templates, but these are randomly placed in time and transformed, as in loops. The goal is to facilitate the synthesis of a composition with many repeating components, without specifying precisely when each event occurs. The real-time parameters for controlling this also enable the tone of a piece to change over time while using the same set of components, simply by synthesizing these components differently.

When a template is added to a mixed bag, it can be set to play either once or repeatedly. It also has a "likelihood" parameter, which determines the probability of that

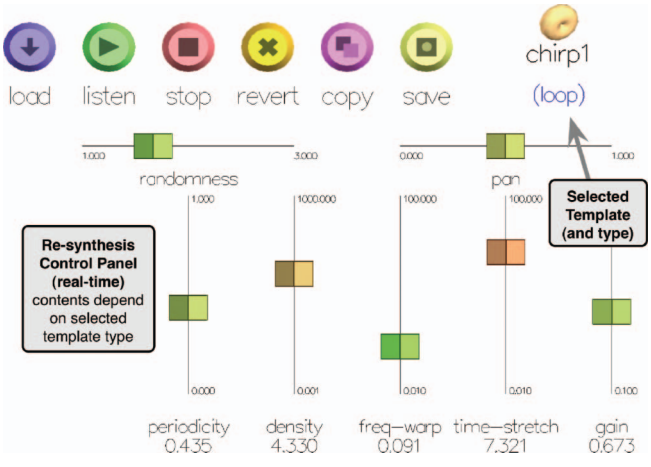

Fig. 10. Sliders for controlling an event loop.

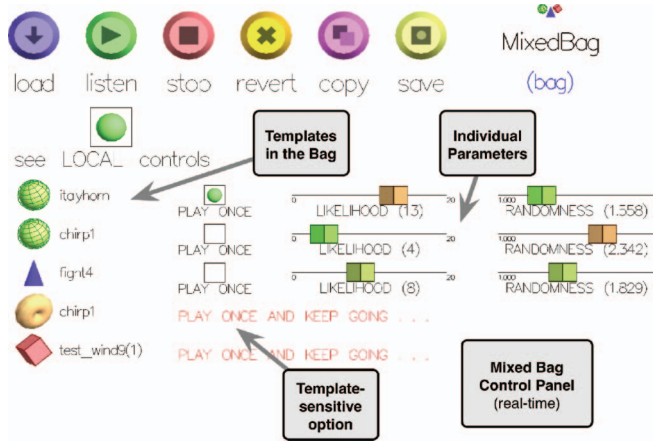

Fig. 11. Sliders for controlling items in a mixed bag.

template's being played in preference over any of the other templates in the bag. Finally, it has a "randomness" parameter, which controls the range for random transformations on that template, analogous to the randomness control in event loops.

Beyond these individual template parameters, each mixed bag has overall periodicity and density settings, which control the temporal distribution of repeating templates in the same way that an event loop does. However, while an event loop plays instances of a single event, a mixed bag randomly selects a repeating template from its list whenever it is time to synthesize a new instance. Templates with higher likelihood settings are more likely to be selected for synthesis.

One way to think of a mixed bag is as a physical bag of marbles. The overall periodicity and density parameters determine when and how often someone dips his hand in the bag and pulls out a marble, or a template to be synthesized. The likelihood setting of a template or marble controls how likely it is for the hand to pull out that particular marble. A repeating marble is tossed back into the bag as soon as it has been drawn and observed (played).

4.7 Pitch and time quantizations

While sliders control the synthesis parameters in a continuous way, more customized musical control can be exerted by quantizing pitches and times to user-specified values. Pitch and time tables can be loaded on-the-fly for each template.

The frequency scaling factor of a template is quantized to the nearest entry in its pitch table, if it has one. This directly sets the frequency at which a sinusoidal or transient event is synthesized. For event loops and mixed bags, it controls the possible frequency scaling during random transformations on the underlying events. The frequencies of individual templates on a timeline are scaled, in the order in which they are played, by successive entries on the timeline's pitch table. This allows a user-defined musical scale to be applied to most templates.

Rhythm can be similarly specified by quantizing time to the nearest entry in a time table. In event loops and mixed bags, this quantizes the event density parameter as well as the intervals between consecutive events. On timelines, templates are positioned only at time points corresponding to table entries, if a table exists. Thus, templates can start synthesizing at particular beats.

4.8 Score language

The manipulations described so far can be controlled via a visual interface. Even finer control over the synthesis can be obtained through the use of a score language. The audio programming language ChucK (Wang and Cook, 2003) is used here both for specifying precise parameter values and for controlling exactly how these values change over time. Since ChucK allows the user to specify events and actions precisely and concurrently *in time*, it is straightforward to write scores to dynamically evolve a sound tapestry.

A ChucK virtual machine is attached to TAPESTREA, which registers a set of API bindings with which ChucK programs can access and control sound templates and automate tasks. Each script (called a *shred*) can be loaded as a sound template and be played or put on timelines. Scripts can run in parallel, synchronized to each other while controlling different parts of the synthesis. It is also possible to create, from within a script, user interface elements for controlling intermediate variables and events used in the script itself. Further, scripting is an easy way to add "traditional" sound synthesis algorithms as well as real-time control via MIDI and Open Sound Control.

4.9 Other controls

TAPESTREA also offers some generic synthesis and playback controls. The gain and stereo panning of templates can be controlled individually, or randomly set by event loops and mixed bags. A reverb effect adapted from STK (Cook & Scavone, 1999) can also be added to the final synthesized sound.

The synthesis interface provides several ways to instantiate new templates. Any existing template can be copied, while sinusoidal and transient event templates can also be saved as event loops. New timelines and mixed bags can be freely created, and existing templates can be dragged onto or off these as needed. Templates can also be deleted from the library, provided they are not being used in a timeline or a mixed bag. Finally, while sound is generally synthesized in real-time, TAPESTREA offers the option of writing the synthesized sound to file.

5. Discussion

TAPESTREA makes it possible to create a wide range of *musical tapestries*. We describe one example re-composition here. The spectrogram (Figure 12) represents a 5 min improvised piece called *Etude pour un Enfant Seul* (Study for a Child Alone). The source sound templates were extracted from the BBC Sound Effects Library. They include the following: a baby's cry (from CD 27, #6, 69.4 to 70.8 s; extracted as sinusoidal: 5 tracks), a bell (CD 14, #8, 0.5 to 7; sinusoidal: 25 tracks), glass breaking (CD 18, #13, 0.5 to 1.5; sinusoidal: 4 tracks), a horn honk (CD 9, #12, 42.9 to 43.4; sinusoidal: 10 tracks), a bird chirp (CD 12, #11, 19.8 to 20; sinusoidal: 4 tracks), and several battlefield sounds (CD 18, #68, 0.5 to 0.8 and 31 to 31.5; CD 18, #69, 1.47 to 1.97 and 31.9 to 32.4; transients). Additional templates, including an

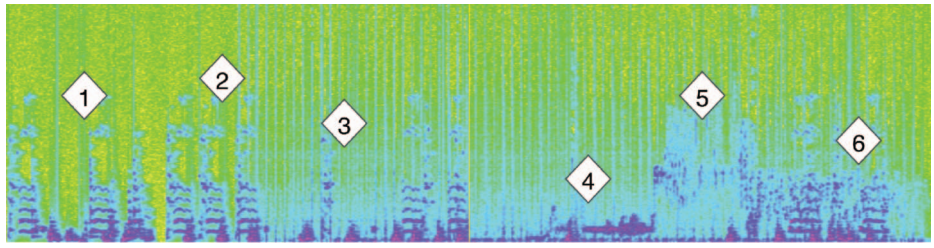

Fig. 12. Example of a soundscape re-composition. Diamonds represent areas of significant shift in the piece.

ocean background with bird chirps removed, were extracted but not used here.

We highlight some areas of interest in the re-composition (denoted by numbered diamonds in Figure 12). In area (1) are time/frequency-warped instances of the baby (7× time-stretch, 0.5× frequency-scaled), horns (6× time, 0.2× and 0.28× freq), and glass (4× time, 0.5× freq). The percussion involving the battlefield transient templates begins around (2) and is dynamically coordinated by scripts. In (3), the percussion develops, punctuated by a solitary glass breaking sound. At (4), greatly modified bird chirps (0.15× time; 0.4× freq) fade in as part of a periodic loop, which is so dense that chirps are triggered at audio rates, forming a rich tone. As time-stretch, frequency-scale, and density are modified, the tone gradually morphs into a flock of birds and back. Combined with further modifications to periodicity and randomness, the flock reaches its peak at (5), modelling the sound of more than 30 birds spread out in time, frequency, volume, and pan – all from a single bird chirp template. The flock is then manipulated to sparser texture, and the child returns at (6) with three longer cries (baby cry; 9× time, 0.4× freq).

Short excerpts from the original recordings, along with extracted templates and the final re-composition, are available online at:

http://taps.cs.princeton.edu/jnmr_sound_examples/

This simple example led to a more complex re-composition, *Etude II pour un Enfant Seul (Loom)*, which was played at the International Computer Music Conference, 2006. A two-channel version of *Etude II* is also available at the above website.

While these examples make good use of TAPESTREA, it is equally possible to create completely differently styled compositions using the same tool and even the same initial sounds.

6. Conclusion

TAPESTREA is a technique and system for "re-composing" recorded sounds by separating them into distinct components and weaving these components into *musical tapestries*. The technique is applicable to musique concrète, soundscape composition and beyond, while the system combines algorithms and interfaces for implementing the concepts. Key contributions include: (1) an approach for re-composing natural sounds, defining semantically clear *sound template* types linked to specific processing techniques, (2) a system for extracting selected sound components into reusable templates, and for transforming and synthesizing these, (3) a class of user interfaces aimed to facilitate the process.

The TAPESTREA interface simultaneously provides visual and audio information, while the system provides the means to interactively extract sound components, transform them radically while maintaining salient features, model them individually or in groups, and synthesize the final multi-level "re-composition" in any number of ways ranging from a pre-set score to dynamically in real-time. Even with a modest set of original sounds, there is no end to the variety of *musical tapestries* one might weave.

Acknowledgements

We are grateful to the extended Princeton Sound Lab family for their help and support.

References

Amatriain, X. & Arumi, P. (2005). Developing cross-platform audio and music applications with the CLAM framework. In: *Proceedings of the International Computer Music Conference*, pp. 403–410.

Bello, J.P., Daudet, L., Abdallah, S., Duxbury, C., Davies, M. & Sandler, M.B. (2005). A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing, 13*(5), 1035–1047.

Bogaards, N., Röbel, A. & Rodet, X. (2004). Sound analysis and processing with AudioSculpt 2. In: *Proceedings of the International Computer Music Conference*, Miami, USA.

Cook, P.R. & Scavone, G.P. (1999). The Synthesis ToolKit (STK). In: *Proceedings of the International Computer Music Conference*, Beijing, China.

Dhomont, F. (1995). Acousmatic update. *Contact!, 8*(2).

Dolson, M.B. (1986). The phase vocoder: a tutorial. *Computer Music Journal, 10*(4), 14–27.

Dubnov, S., Bar-Joseph, Z., El-Yaniv, R., Lischinski, D. & Werman, M. (2002). Synthesizing sound textures through wavelet tree learning. *IEEE Computer Graphics and Applications, 22*(4), 38–48.

Ellis, D.P.W. (1994). A computer implementation of psychoacoustic grouping rules. In: *Proceedings of the 12th International Conference on Pattern Recognition*, Jerusalem, Israel, pp. 108–112.

Klingbeil, M. (2005). Software for spectral analysis, editing, and synthesis. In: *Proceedings of the International Computer Music Conference*, Barcelona, Spain, pp. 107–110.

McAulay, R.J. & Quatieri, T.F. (1986). Speech analysis/synthesis based on a sinusoidal representation. *IEEE Transactions on Acoustics, Speech, and Signal Processing, 34*(4), 744–754.

Melih, K. & Gonzalez, R. (2000). Source segmentation for structured audio. In: *Proceedings of IEEE International Conference on Multimedia and Expo (II)*, New York, USA, pp. 811–814.

Roads, C. (2002). *Microsound*. Cambridge: MIT Press.

Rudy, P. (2004). Spectromorphology hits hollywood: black hawk down – a case study. In: *Proceedings of the International Computer Music Conference*, pp. 658–663.

Schaeffer, P. (1950). Introduction à la musique concrète. *La Musique Mécanisée: Polyphonie, 6*, 30–52.

Schaeffer, P. (1952). *À la recherche d'une musique concrète*. Paris: Seuil.

Schafer, R.M. (1977). *The tuning of the world*. New York: Knopf.

Serra, X. (1989). A system for sound analysis/transformation/synthesis based on a deterministic plus stochastic decomposition. PhD thesis, Stanford University, USA.

Truax, B. (1990). Composing with real-time granular sound. *Perspectives of New Music, 28*(2).

Truax, B. (2002). Genres and techniques of soundscape composition as developed at Simon Fraser University. *Organised Sound, 7*(1), 5–14.

Verma, T.S. & Meng, T.H. (1998). An analysis/synthesis tool for transient signals that allows a flexible sines+transients+noise model for audio. In: *Proceedings of 1998 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Seattle, USA, pp. 3573–3576.

"Voice of the beast – The sounds of *The Perfect Storm*." (2000). DVD-ROM Featurette on DVD release of *The Perfect Storm*, Warner Home Video. Also on WarnerVideo.com website: http://warnervideo.com/perfectstormevents/popup/video/voice_300.html

Wang, G. & Cook, P.R. (2003). ChucK: a concurrent, on-the-fly, audio programming language. In: *Proceedings of the International Computer Music Conference*, Singapore, pp. 219–226.