Audio and User Directed Sound Synthesis

Marc Cardle, Stephen Brooks, Peter Robinson
Computer Laboratory, *University of Cambridge*
email: {mpc33, sb329, pr}@cl.cam.ac.uk

Abstract

We present techniques to simplify the production of soundtracks in video by re-targeting existing soundtracks. The source audio is analyzed and segmented into smaller chunks, or clips, which are then used to generate statistically similar variants of the original audio to fit particular constraints. These constraints are specified explicitly by the user in the form of large-scale properties of the sound texture. For instance, by specifying where preferred clips from the source audio should be favored during the synthesis, or by defining the preferred audio properties (e.g. pitch, volume) at each instant in the new soundtrack. Alternatively, audio-driven synthesis is supported by matching certain audio properties of the generated sound texture to that of another soundtrack.

1 Introduction

Human perception of scenes in the real world is assisted by sound as well as vision, so effective videos require the correct association of sound and visuals. Currently, artists are faced with the daunting task of finding, recording or generating appropriate sound effects and ambiences, and then fastidiously arranging them to fit the video, or changing the video to fit the soundtrack.

We present a solution for simple and quick soundtrack creation that generates new, controlled variations on the original sound source of arbitrary length, which still bear a strong resemblance to the original, using a controlled stochastic algorithm.

The idea for this work is inspired by recent work on sound textures [1,3,4]. In Dubnov et al. [1], the conditional probabilities of the paths of the input sound's wavelet tree representation is learnt and sampled to generate new random tree instances. Hoskinson [3] and Lu et al. [4] operate in a similar fashion by first analyzing and segmenting the input into variable-sized chunks that are recombined into a continuous stream, where each chunk is statistically dependant on its predecessor. In all these previous approaches, no control is possible over new instances of a sound texture since they are by definition random. By adding user control over the synthesis process, the basic concept of a sound texture can be extended to further increase its applicability. Based on Lu et al.'s self-similarity-based approach [4], our controllable sound textures allows for various types of user interactivity as described below.

In the simplest case, users manually indicate large-scale properties of the new sound to fit an arbitrary video. This is done by manually specifying which types of sounds in the original audio are to appear where in the new soundtrack. A controllable statistical model is extracted from the original soundtrack and a new sound instance is generated that best fits the user constraints.

It is also possible to use audio to constrain the synthesis process. The goal here is to produce a new sound texture that exhibits some similar property (such as volume or pitch) to that of a separate guiding soundtrack. For example, a laughing audience's recording could be automatically replaced by a synchronized 'booing' soundtrack. The user only has to specify the audio matching feature to use.

An advantage of our method is that it provides a very natural means of specifying soundtracks. Rather than creating a soundtrack from scratch, broad user specifications such as '*more of this sound and less of that sound*' are possible. Alternatively, a user can simply supply a driving soundtrack and, given a new target sound, say, in effect: '*Make it sound like this whilst syncing with that*'. This significantly simplifies existing soundtrack recycling since no editing, wearisome looping, re-mixing or imprecise sound source separation is necessary.

We first describe our unconstrained sound texture model and then extend it to provide control. After which, three interaction methods are outlined.

2 Random Sound Synthesis

For the sake of brevity, only a short description of Lu et al.'s two-stage sound synthesis [4] is given here. In the analysis stage, the similarity matrix M is first derived by calculating the difference in Mel-Frequency Cepstral Coefficients (MFCCs) from every frame of the input audio to every other. By sliding a cross-correlation kernel [2] along the diagonal of M, we obtain the self-similarity novelty curve N for the input. The input is then segmented into shorter clips along the local maxima of N. These points correspond to points of maximum audio change such as pattern breakpoints. The resultant clips form the input's characteristic building patterns.

The sound texture is essentially a Markov process, with each state corresponding to a single clip, and the probabilities P_{ij} corresponding to the likelihood of transitions from one clip i to another j. The transition

probability from frame *i* to frame *j* depends on the MFCC similarity between frames *i*+1 and *j* so that:

$$P_{ij} \propto e^{\left(\frac{S_{i+1,j}-1}{\sigma}\right)} \quad \text{with} \quad S_{ij} = \sum_{k=-m}^{m} w_k \left(\frac{V_{i+k} \bullet V_{j+k}}{\|V_{i+k}\| \cdot \|V_{j+k}\|} \right)$$

where σ is a scaling parameter and S_{ij} is the similarity distance between frame *i* and *j*. S_{ij} is defined as the weighted sum of the autocorrelation vectors over the previous *m* and next *m* neighbouring temporal frames with binomial weights $[-w_m,\ldots,w_m]$, where V_i and V_j are the MFCC feature vectors of frames *i* and *j*. All the probabilities for a given row of P are normalized so that $\Sigma_j P_{ij} = 1$. The synthesis step, or *random play*, does a Monte-Carlo sampling of P to decide which clip should be played after a given clip.

3 Directed Sound Synthesis

The above algorithm works well with almost no artifacts on both stochastic and periodic sound textures. However, no control is possible over new instances of a sound texture since they are by definition random. We now introduce high-level user-control over the synthesis process. This is achieved by enabling the user to specify which types of sounds from the input sound should occur when, and for how long, in the output synthesized soundrack. These user-preferences translate into either hard or soft constraints during synthesis. In this section, we first look at how these synthesis constraints are defined, and then, by what means they are enforced in our controlled algorithm.

3.1. Constraint Specification

In order to synthesize points of interest in the soundtrack, the user must identify the synthesis constraints. First, the user selects a source segment in the sample sound such as an explosion in a battle soundtrack. Secondly, the user specifies a target segment indicating when, and for how long, in the synthesized sound the explosion(s) can be heard. The constraints for the rest of the new soundtrack can be left unspecified, so that in our video example, a battle-like sound ambience will surround the constrained explosion.

The source and target segments, each defined by a start and end time, are directly specified by the user on a familiar graphical *amplitude* x *time* sound representation. Since the target soundtrack has yet to be synthesized and therefore no amplitude information is available, target segments are selected on a blank amplitude timeline of the length of the intended sound. Note that the number, length and combinations of source and target segments are unrestricted, and that exclusion constraints can also be specified so as to prevent certain sounds from occurring at specific locations.

The user can associate a probability with each constraint, so controlling its influence on the final sound. To this end, a weighting curve is assigned to each target segment, designating the probability of its associated source segment(s) occurring at every point in the target area. The weights vary from [-1, 1], where -1 and 1 are equivalent to hard-constraints guaranteeing, respectively, exclusion or inclusion. Soft-constraints are defined in the weight ranges (-1,0) and (0,1) specifying the degree with which exclusion or inclusion, respectively, is enforced. Furthermore, the reserved weight of 0 corresponds to unconstrained synthesis. The weights of overlapping targets are added up so that clips that satisfy both constraints are even more or less likely to occur. For consistency, the user is prevented from defining overlapping hard-constraints.

Therefore, each separate constraint $c \in 1,\ldots,n$ defines one or more source segments S_c from the input sound, one or more target segments T_c in the new audio and W_c, the weighting curve defining the likelihood of audio from S_c appearing at every instant of T_c. Once the input audio has been segmented, the clips included in time segments S_c form s_c, the set of source clips for constraint *c*.

3.2. Hard and Soft Constrained Synthesis

Directed synthesis is achieved by dynamically scaling the clip probabilities P_{ij} in the Markov table during synthesis so as to maximize constraint satisfaction. Since clips originating from the current targeted source are preferred, we therefore proportionally increase their associated likelihood of being selected. Let *i* be the last generated clip and *u* the current temporal synthesis position, then for all $\boldsymbol{c} \in \{\boldsymbol{c} \in 1,\ldots,n \mid u \in T_{\boldsymbol{c}}\}$, we must rescale all the probabilities P_{ij} to the next clip *j* for all $j \in s_{\boldsymbol{c}}$. In other words, when synthesizing inside a constrained target segment, we rescale the probabilities of all clips belonging to s_c in the following manner:

$$P'_{ij} = \max\left(\min\left(\left(f(P_{ij}) w_u + 1 \right) P_{ij}, 0.999 \right), 0 \right) \quad (1)$$

where $w_u = \max(\min(\Sigma_{\boldsymbol{c}} W_{\boldsymbol{c}}, -1), 1)$ so that weights of overlapping constraints are added up. w_u and P'_{ij} are clamped to respectively prevent illegal weight and probability values. The scaling function *f* determines the influence of the constraint weights whilst favoring greater scaling on lower weights, and inversely. *f* is defined as:

$$f = e^{\left(\frac{1}{P_{ij}+k}\right)} - e^{\left(\frac{1}{1+k}\right)} + 1 \qquad (2)$$

where k controls the overall weight influence in the synthesis and is user-settable. Bigger values of k, better enforces the constraints at the cost of randomness and audio continuity.

We then rescale and renormalize the weights of all other unconstrained clips in P_i so that they share the left-over probabilities from the scaled constrained weights. Let $m \notin s_c$ then $P'_{im} = (1-\Sigma_j P'_{ij})(P_{im}/\Sigma_m P_{im})$.

- If $\Sigma_j P'_{ij} > 1$ (1) or if $w_u \geq 1$ (2), then for $\forall m \notin s_c$, $P'_{im} = 0$ and $P'_{ij} = P'_{ij}/\Sigma_j P'_{ij}$.
- If $\exists y \in \{y \in \mathbf{c} \mid W_y = 1\}$ (3), then for $\forall m \notin s_y$, $P'_{im} = 0$.
- If $\exists y \in \{y \in \mathbf{c} \mid W_y = -1\}$ (4), then $\forall m \in s_y$, $P'_{im} = 0$.
- If $w_u \leq -1$ (5), then $\forall m \notin s_c$, $P'_{im} = 0$.

Conditions (2) and (3) ensure that any detected hard-constrained weights equal to 1 provoke the exclusion of clips from a different source. In a similar fashion, condition (4) and (5) exclude any sources with hard-constrained weights equal to –1.

3.3 Anticipation

Smooth transitions near boundaries between unconstrained and hard-constrained areas might sometimes be difficult to achieve. This is due to the fact that hard-constrained areas drastically reduce the clip candidate set, potentially forcing the selection of clips with low probabilities when jumping from unconstrained to hard-constrained areas and vice-versa. Adding hard-constraints anticipation capabilities to the synthesis avoids such situations. In this manner, a look-ahead mechanism is triggered when the current synthesis position u below a distance d from the start of a hard-constrained area T_c such that:

$$d = \left\lceil \frac{s_{sound} \times \beta}{s_{clip}} \right\rceil \times s_{clip} \quad (3) \quad \text{and} \quad s_{clip} = \frac{s_{sound}}{n_{clip}} \quad (4)$$

where s_{sound} is the total length of the input sound, n_{clip} the number of clips detected during segmentation, s_{clip} the average clip size and β controls the anticipation distance (usually set to 5%).

Anticipation works by backtracking from the start of T_c up until u with a step-size of s_{clip} and using the gathered data to bias the clip choice at point u. The effect is that, gradually, clips with distant successors in s_c will be favored. For example, before synthesizing the last unconstrained clip i before T_c, we first find the top n clips a_1 with the highest likelihood that their next clip j belongs to s_c. Hence, clips a_1 are favored by increasing their respective probabilities when picking clip i. We then backtrack by s_{clip} and find the top n clips a_2 with the highest likelihood that their next clip belongs to a_2. This continues r-times until we reach u where a_r is used to bias the choice of the next clip. Before renormalization, the probabilities of the top n clips a_p are scaled in the following manner:

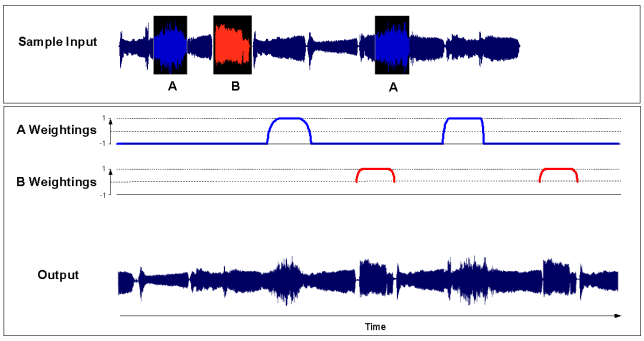

Figure 1 (*Top*) Source regions A and B. (*Middle*) Weighting curve for A and B. (*Bottom*) Directed synthesis output.

$$P'_{ij} = P_{ij} + \alpha P_{ij} \left(\frac{p \times s_{clip}}{d} \right) \qquad (5)$$

where α determines the enforcement level of the anticipation (usually set to 30%).

4 User Control

In this section, we look at how users specify the synthesis constraints in the manual interaction mode. The user starts by specifying one or more source regions in the sample sound. In the example depicted in Figure 1, two distinct source regions are defined corresponding to areas A and B (top). Note that A is defined by two segments. The user then draws the target probability curve for both sources A and B directly on the timeline of the new sound. A's weightings are zero except for two sections where short and smooth soft-constraints lead to a 1-valued hard-constraint plateau. This results in region A smoothly appearing twice, and nowhere else. On the other hand, B's curve also defines two occurrences but is undefined elsewhere, imposing no restrictions. Thus sounds from B might be heard elsewhere.

In this example, both sounds comprising source A are similar. Instead of selecting both sounds, the user can simply select the first segment of A and then let the system find similar sounds. By performing *sound-spotting* audio matching [5], perceptually similar audio segments to the sound A are found in the rest of the soundtrack. This is especially valuable for selecting frequently recurring sounds over extended soundtracks.

5 Audio-driven Sound Synthesis

The goal here is to use audio as the synthesis constraint so as to produce a new sound texture that exhibits some similar property (such as volume or pitch) to that of a separate guiding soundtrack X_{guide}. Up until now, to synthesize a new sound X_{new}, a source sound X_{source} and a set of user-constraints were required. Instead here, we replace the soundtrack X_{guide} with another one X_{new}, built up from sounds in X_{source}, by simply matching some feature of the old soundtrack X_{guide} with that of the new soundtrack X_{new}. That way, we can change the building blocks of a soundtrack, without changing its overall properties.

Let i be the last generated clip and u the current temporal synthesis position in X_{new}, then we must rescale all the probabilities P_{ij} to the next clip j so as to maximize the likelihood that clip j will exhibit similar audio properties to that of X_{guide} at the same position u.

The user first defines the audio feature, or combination of audio features, to use for the matching. Examples of which are the mel-frequency cepstral coefficients, RMS volume, short time energy, zero crossing rates, sub-band powers distribution, brightness, bandwidth or spectrum flux. These are then pre-calculated for every sliding window position in X_{source} previously used in the segmentation algorithm. The same is also carried out over X_{guide}. These features are used to calculate the distance D_j^u between all the windows from X_{source} forming the potential next clip j and the corresponding windows in X_{guide}:

$$D_j^u = \frac{\sum_{n=0}^{t_j} \left(f\left(w_{u+n\times wsize}^{X_{guide}}\right) - f\left(w_{b_j+n\times wsize}^{X_{source}}\right)\right)^2}{t_j} \qquad (6)$$

where $w_u^{X_{guide}}$ is the window from X_{guide} at position u, $w_u^{X_{source}}$ the window from X_{source} at position u, s_j the total number of windows forming clip j, f the feature extraction routine and the *wsize* the window sample size (e.g. 1024). Before synthesizing each new clip, D is evaluated for all potential next clips and its renormalized value is used as weight w_u in Equation (1). The synthesis then proceeds as normal with these rescaled weights.

6 Results and Discussion

The examples are in the accompanying video[1] as printed figures could not convey our results meaningfully. The first example illustrates the use of manual control to derive a new sound track from an existing one when the corresponding video sequence is edited. If the results are not what the user expected, at most a small number of iterations are required to produce a soundtrack that better accommodates the user's intentions. This is relatively quick since synthesis is done in real-time. Not surprisingly, better results are obtained if the source and targets regions are similar in length, otherwise unexpected results occur. For example, a laughing sequence sounds unnatural if it is prolonged for too long using hard-constraints.

Our second example illustrates the use of audio-driven synthesis. Given a racing video, the soundtrack of a dragster-like motor is automatically replaced by that of lawnmower's. We simply use the RMS volume as the matching feature. In our final example, we use voice to drive the synthesis. The user simply records himself imitating the sound from X_{source} that he/she wants at any given point in X_{new}. The recording is then used as X_{guide} in the synthesis. This is a rapid way of generating controlled sound textures using an intuitive, rapid and familiar interface.

Note that it is possible for the user to simply draw one or more curves defining the values $f(w_{1:end}^{X_{guide}})$ in Equation (6). In this manner, a user-specified pitch curve could be directly used to control the synthesized.

7 Conclusion and Future Work

We introduce a new method for generating controllable sound textures. The user is given several intuitive ways of directing the synthesis process through manual constraint specification, soundtrack-driven and voice-driven synthesis as well as preferred audio properties curves.

There are still many opportunities for future work such as identifying and dynamically resizing quasi-silent portions [6] in the clips, as well as time-scaling [4] them, so as to better fit the user constraints. It would also be useful to develop a more intuitive way of defining multi-dimensional preferred audio properties curves.

References

[1] DUBNOV, S., BAR-JOSEPH, Z., EL-YANIV, R., LISCHINSKI, D., AND WERMAN, M. 2002. Synthesizing sound textures through wavelet tree learning. *IEEE Computer Graphics and Applications.*

[2] FOOTE, J. AND COOPER, M. 2001. Visualizing Musical Structure and Rhythm via Self-Similarity. *Proc. International Conference on Computer Music (ICMC 2001)*, Habana, Cuba, September 2001.

[3] HOSKINSON, R., AND PAI, D., 2001. Manipulation and Resynthesis with Natural Grains. *Proc. of the International Computer Music Conference.*

[4] LU, L., LI, S., LIU, W., AND ZHANG, H.. 2002. Audio Textures. *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing.*

[5] SPEVAK, C., AND POLFREMAN, R., 2001. Sound spotting - A frame-based approach, *Proc. of the Second Annual International Symposium on Music Information Retrieval: ISMIR 2001.*

[6] TADAMURA, K., AND NAKAMAE, E. 1998. Synchronizing Computer Graphics Animation and Audio, *IEEE Multimedia*, Oct-Dec. No 2, Vol 5.

[1] Video: http://www.cl.cam.ac.uk/users/mpc33/icmc03.html