Audio Textures: Theory and Applications

Lie Lu, Liu Wenyin, *Senior Member, IEEE*, and Hong-Jiang Zhang, *Fellow, IEEE*

***Abstract*—In this paper, we introduce a new audio medium, called *audio texture*, as a means of synthesizing long audio stream according to a given short example audio clip. The example clip is first analyzed to extract its basic building patterns. An audio stream of arbitrary length is then synthesized using a sequence of extracted building patterns. The patterns can be varied in the synthesis process to add variations to the generated sound to avoid simple repetition. Audio textures are useful in applications such as background music, lullabies, game music, and screen saver sounds. We also extend this idea to audio texture restoration, or constrained audio texture synthesis for restoring the missing part in an audio clip. It is also useful in many applications such as error concealment for audio/music delivery with packets loss on the Internet. Novel methods are proposed for unconstrained and constrained audio texture synthesis. Preliminary results are provided for evaluation.**

***Index Terms*—Audio texture restoration, audio texture synthesis, audio textures, constrained audio texture.**

I. Introduction

THE size of audio media is an important consideration in applications involving audio. The concerns include the storage needed for the audio data, and the time needed for download and transmission when the Internet is involved, especially for the narrow-band network environments. How to make such media objects small in sizes will be critical to the success of the applications.

In many applications, there is a need for a simple sound of arbitrary length, such as lullabies, game music, and background music in screen savers. Such sounds are relatively monotonic, simple in structure, and characterize repeated yet possibly variable sound patterns. A very long, simple but not exactly repeating sound would require huge storage. It will be better if we have a technology to generate such a long audio stream from a given short and simple audio clip. Thus we can only store the short audio clip, and then generate a long audio stream of any length in the user end. By doing so, much storage and transmission time can be saved.

In this paper, we introduce the idea of a new audio media, which is referred to as audio texture, as an efficient method for generating such long sounds from example clips. We call it audio texture because it exhibits repeated or similar patterns, just like image textures and video textures [1]. Audio texture provides an efficient means of synthesizing continuous, perceptually meaningful, yet nonrepetitive audio stream from an example audio clip. It is "perceptually meaningful" in the sense that the synthesized audio stream is perceptually similar to the given example clip. However, an audio texture is not just a simple repetition of the audio patterns contained in the input; variations of the original patterns are fused into it to give a more vivid stream. The audio stream can be of arbitrary length according to the need.

The idea of audio texture is inspired by video textures [1], a new type of visual medium. The latter was proposed as a temporal extension of two-dimensional (2-D) image texture synthesis [2], [3], and is researched in the areas of computer vision and graphics. It is natural to generalize the idea to audio data. Audio data as a signal sequence presents self-similarity as a video sequence does. The self-similarity of music and audio has been shown in [4] using a visualization method. So far, audio similarity is mostly studied for audio or music classification and retrieval only [5], [6] but not for new audio clip generation. Musical Mosaicing [18] addressed an issue of retrieving sound samples in large database and combine them to generate a new sequence, by specifying only high-level properties. It is similar to audio texture on generating a new sequence. However, musical mosaicing is generated from a corps of sound samples, while audio texture is generated based on the self-similarity of a given audio clip.

One kind of audio texture synthesis is to generate, from a short piece of example audio clip, an arbitrarily long audio sequence which bears similarity patterns to the original clip yet presents variations. In such a case, audio texture is generated from the audio example relatively freely, since there are no special constraints on it. We refer to this kind of audio texture synthesis as unconstrained audio texture synthesis.

We also extend the idea to constrained texture synthesis, in which a part of audio signal is lost because of some reasons, as illustrated in Fig. 1, where the missing part is assumed as being filled with zero values. In this case, audio texture can be used to restore the missing part according to the self-similarity of audio structure. Such audio texture synthesis is constrained by the beginning and ending points of the missing part. That is, the synthesized part should be perceptually smooth at the joint points with the remaining audio clip. No uncomfortable break or click is expected at those points. Hence, we refer to this kind of restoration as constrained audio texture synthesis, compared to the former unconstrained audio texture synthesis which is free from the boundary effects. Since the constrained audio texture synthesis generates a frame sequence which can be used to perceptually smoothly restore the missing part, it is also called audio texture restoration. It is a little bit different from the traditional digital audio restoration, where the objective is

Manuscript received November 22, 2002; revised August 26, 2003. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. C.-C. Jay Kuo.

L. Lu and H-J. Zhang are with Microsoft Research Asia, Beijing, 100080, China (e-mail: llu@microsoft.com; hjzhang@microsoft.com).

L. Wenyin was with Microsoft Research Asia, Beijing, 100080, China. He is now with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: csliuwy@cityu.edu.hk).

Digital Object Identifier 10.1109/TSA.2003.819947

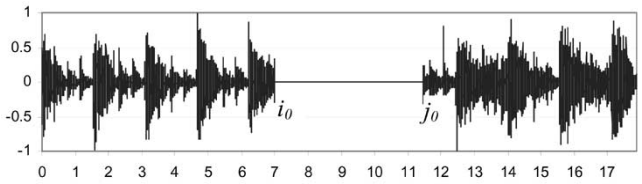

Fig. 1. Audio example which lost many frames.

not to restore the missing part, but to restore the degraded audio signals, such as click removal, noise removal for gramophone recordings, film sound tracks, and tape recordings [13].

Constrained audio texture synthesis is also very helpful in some other applications. For example, in the application of audio/music delivery on the internet, some missing frames can be caused due to lost packets, audio textures synthesis can be used to restore these missing frames. Compared with most traditional error concealment methods, which only dealt with errors with a short length (typically around 20 ms or several packets) [11], [12], our method can restore the audio with loss of a much longer length, such as 1 s and more.

A two-stage method, which includes analysis and synthesis, is proposed for generating audio textures with or without constraints. In the analysis stage, the example clip is analyzed in structure, and segmented into sub-clips by extracting its building patterns or equivalently finding pattern breakpoints. This step is based on the similarity measure between each pair of frames according to their Mel-frequency Cepstral Coefficients (MFCCs). In the synthesis stage, the sequence of the sub-clips or frames is decided for generating new audio stream in unconstrained case or restoring the missing audio part in constrained case. Variable effects can be combined into the building patterns to avoid monotony of the newly synthesized audio stream.

However, it should be noted that the proposed audio texture synthesis method is currently more suitable for those audio signals with simple structures. For simple structure audio, it is relatively easy to analyze its structure and extract its building pattern more accurately, and hence, will make the generated or restored audio perceptual better. It may be not applicable to complex music such as arbitrary music pieces from compact disks. For example, for those songs or music with lyrics, the method will fail. It is difficult to generate perceptually smooth and meaningful lyrics only based on self similarity.

The rest of the paper is organized as follows. Section II presents an overview of the proposed methods for audio textures generation with or without constraints. Section III discusses the algorithms for analyzing audio structure. Section IV and Section V describes the algorithms for unconstrained audio texture synthesis and constrained audio texture synthesis, respectively. Section VI presents some applications on audio textures and provides some preliminary results. Conclusions are given in the Section VII.

II. System Overview

The proposed method for audio texture generation (both constrained and un-constrained) can be divided into two stages: analysis and synthesis, as shown in Fig. 2.

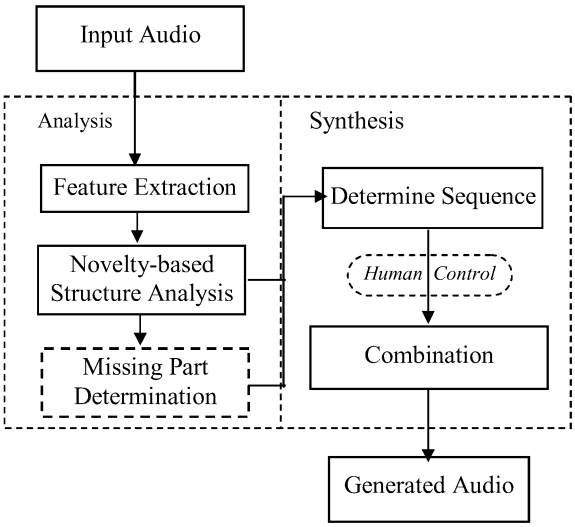

Fig. 2. System overview diagram on unconstrained and constrained audio textures syntheses, in which the solid rectangles and solid arrows represent the basic synthesis process; the dashed rectangle means that it is used in the constrained case but unnecessary in the unconstrained case; and the dashed ellipse means that user control can be introduced for final combination.

In the analysis stage, feature is extracted to represent the original audio data. The main feature used in our approach is MFCC. MFCC is a set of perception-based spectral features. It is commonly used with great success in speech and audio processing [4], [6], [14]. Actually, many other features could also be used for such purpose, such as Linear Predictive Coding (LPC) [14], correlogram [15], and others special features [5]. However, although these features are useful in some specific applications, they are not proved significantly better or more universal than MFCC for general audio applications. Hence, we use only MFCC in our implementation in this paper.

Then, the structure of the audio clip is analyzed, and the audio clip is segmented into several basic building patterns or sub-clips, where a pattern or sub-clip can be composed of a single frame or multiple frames. From many algorithms for audio segmentation [7], [16], [17], a novelty detection method [7] is selected and used in our approach to segment the audio clip into sub-clips. Meanwhile, the similarity and transition probability between each two sub-clips are calculated for further synthesis. In the case of constrained synthesis, the missing part of audio clip should be located firstly. In Fig. 2, the module of missing part determination is drawn as a dashed rectangle, which means it is used in the constrained case but unnecessary in the unconstrained case.

In the synthesis stage, we use sub-clip as the synthesis unit. We still keep using frame as synthesis unit, especially when no obvious building patterns are extracted from the input audio example. Using different synthesis unit is more efficient for different kind of audio and various synthesis requirements. Actually, frames can be considered as a special case of sub-clips.

A sub-clip sequence is first generated based on the transition probabilities, by deciding which sub-clip should be played after a given sub-clip. The sub-sequence determination is free from the boundary constraints in an unconstrained case; while the boundary smoothness should always be kept in mind in a constrained case. Different effects can be introduced by

determining different sub-clip sequence or adding various effects to the sub-clips. The variations include time scaling and pitch shifting, which can be implemented by the synchronous overlap-add (SOLA) method [9], [10]. Users can also indicate what effects should be added and how to combine the sub-clips. The user control module is represented as a dashed ellipse in Fig. 2. Once these are done, a perceptually natural audio stream, or an audio texture, is generated or restored.

III. Analysis Process

In this stage, the structure of the input audio clip is analyzed. It consists of two steps: similarity and transition probability are first measured between each pair of frames, and the audio clip is then segmented into sub-clips as basic building blocks.

The input audio clip is firstly segmented into frames with a uniform length. The uniform frame length is 32 ms in our current implementation. Thus, each frame comprises 256 points for an audio clip with an 8 KHz sampling rate. Discrete Fourier Transform (DFT) is performed on each frame and then 16-order MFCC is calculated from the DFT coefficients

A. Similarity Measure and Transition Probability

In order to generate a perceptually natural audio texture, it is necessary to consider the similarity between any two frames and the transition probability from one to the other. The similarity measure will be used to calculate the novelty score [7], [8], extract the audio structure and segment the original audio into sub-clips. It is also the basis for synthesis if frame is used as the synthesis unit.

Let $\boldsymbol{V}_i$ and $\boldsymbol{V}_j$ be the feature vectors of frames i and j in the MFCC feature space. The similarity measurement is simply based on vector correlation and defined as

$$s_{ij} = \frac{\boldsymbol{V}_i \bullet \boldsymbol{V}_j}{\|\boldsymbol{V}_i\| \cdot \|\boldsymbol{V}_j\|} \tag{1}$$

where S_{ij} represents the similarity between frame i and frame j. Since the feature value is arbitrary in the MFCC space, the values of S_{ij} range in $[-1, 1]$ in theory.

The above measure considers two isolated frames only. In order to give a more comprehensive representation of the similarity, it will be better if their neighboring temporal frames are taken into considerations. Suppose that the previous m and next m frames are considered with weights $[w_{-m}, \ldots, w_m]$, the better similarity is developed as follows:

$$s'_{ij} = \sum_{k=-m}^{m} w_k S_{i+k,j+k}. \tag{2}$$

This method captures the time dependence of the frames. To yield a high similarity score, it requires that the two subsequences should be similar. In this way, we are actually matching two sub-clips instead of just two frames.

In our implementation, a symmetric rectangle weighting window is used, supposing that each frame has the same importance. We also tried other kinds of window functions, such as Hanning window, the results are similar. The length of the window is also difficult to set. Different sounds might need different length of window. For example, given a music clip, if the length of the window is synchronous with its beat, the results might be better. However, since there are no obvious directions on it, a constant length is used for each audio clip, in our current implementation.

Based on the similarity measure, transition probability is defined as follows. The transition probability from frame i to frame j depends on the similarity between frames $i+1$ and j. The more similar these two frames are, the higher the transition probability should be. In this principle, the transition probability is related to the similarity by the following exponential function:

$$P_{ij} = A \exp\left(\frac{S'_{i+1,j-1}}{\sigma}\right) \tag{3}$$

where A is the normalizing constant such that $\sum_j P_{ij} = 1$, and σ is the scaling parameter. Smaller values of σ emphasize the very best transitions while larger values of σ allow for greater variety at the cost of poorer transitions.

Fig. 3 shows an example of the similarity matrix, using 2-D gray images representing s'_{ij} for all i, j, computed from a piece of music clip, which comprises 550 frames. The brightness of a pixel is proportional to the corresponding value. The brighter the pixel is, the larger the similarity is. The transition probability matrix is similar to the similarity matrix but with a one-pixel offset. The figure also illustrates the self-similarity of audio or music structure.

B. Sub-Clip Extraction

Similar to the di-phones in text-to-speech system [9], [21] or the grains in granular synthesis [19], some possible building patterns are also detected from the given audio clip and are used to synthesize textures instead of frames. Building patterns will be obtained by segmenting the input audio clip into sub-clips at some breakpoints. Although many algorithms on audio segmentation [7], [16], [17] exists, a novelty-based method is utilized here. Novelty score [7], [8] is used to measure the possibility of a new building pattern appearing. Segmentation is performed based on the novelty score at each time slot.

Consider a simple audio clip having only two extremely different sound objects, where each object composes of N frames and exists steadily in its lifecycle. Suppose the characteristics of these two objects are totally reversed, the correlation of each object itself is positive one, and the correlation between these two objects is negative one. Thus, the similarity matrix of this simple audio clip is something like the following:

$$S = \begin{bmatrix} I_N & -I_N \\ -I_N & I_N \end{bmatrix} \tag{4}$$

where I_N is a $N \times N$ unit matrix. The diagonal unit matrix corresponds to the sound objects which have high self-similarity, while the off-diagonal matrix corresponds to the low cross-similarity between these two sound objects. It should be noted that these two sound objects are supposed in an extreme case; there may not exist such sound objects that are completely negatively correlated in real world.

If S is correlated with a kernel matrix which looks like S but has a smaller dimension, a maximum value will be obtained at

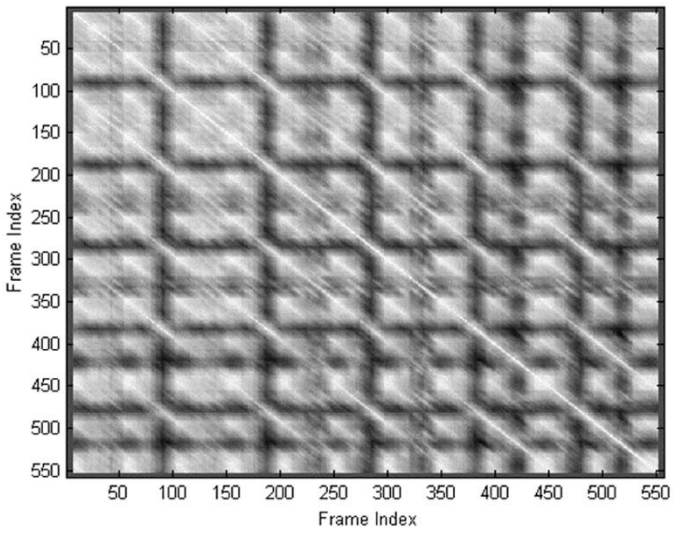

Fig. 3. Similarity matrix of an example music clip.

the boundary of the two sound objects. For example, a simplest kernel matrix can be

$$K = \frac{1}{4}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}. \tag{5}$$

When S is correlated with K along the diagonal direction, the result value is 1 only at the position N, while it is 0 at other positions. That means the boundary is at time slot N. It is similar to the edge detection in image processing.

The correlation value at each diagonal point is taken as the novelty score at that time. In general, the novelty score at the ith frame can be calculated as

$$N_{(i)} = \sum_{m=-w/2}^{w/2} \sum_{n=-w/2}^{w/2} K_{m,n} S_{i+m,i+n} \tag{6}$$

where K is a kernel matrix and S is the similarity matrix. The K in (5) is the simplest kernel and the S in (4) can be also considered as a simple kernel. We can also use a 2-D window function (such as Hanning) to replace the unit matrix in (4) to obtain a new kernel. This kind kernel can avoid edge effects because it tapers toward zero at the edges. In our implementation, 2-D Hanning window function is used to replace the unit matrix.

According to the novelty score, a simple scheme is developed to do sub-clip segmentation: in the novelty curve, the local maxima above a threshold are selected as breakpoints. The sub-clip in each two breakpoints can be considered as a building pattern.

Fig. 4 shows an example of how sub-clips or building patterns can be extracted for a music clip, whose similarity matrix is illustrated in Fig. 3. Fig. 4(a) shows its original music data and Fig. 4(b) shows the corresponding novelty score curve. The local peak is selected as the building pattern boundary. From Fig. 4(b), it could be seen that the local peaks of the novelty curve is basically corresponds to the onsets of the music piece. That is, one note or several notes is extracted as one building pattern. In our experiments, the typical sub-clip length is between 0.3 and 1.2 s.

However, when the given audio clip only contains one sound object, for example, one note or one cry, there will be no obvious peak in the novelty curve. In such case, we take it as one sub-clip. We can also segment it based on the amplitude variation if we want to get more details.

Once sub-clips are extracted, the similarity between each sub-clip should be calculated as basis for further synthesis. Some modification is made on (2) in order to calculate the similarity between each pair of sub-clips, because that definition assumes that sub-clips are of equal length, while the segmented sub-clips using this method are usually of nonequal length. In principle, time-warping and dynamic programming methods should be used to compute the similarity between each pair of sub-clips. However, we used a simplified method as follows.

Suppose sub-clip i contains M frames and begins from the frame i; sub-clip j contains N frames and begins from the frame j; and $M < N$. The similarity between these two sub-clips can be represented by

$$S'_{ij} = \sum_{k=1}^{M} w_k s_{i+k,j+[kN/M]}. \tag{7}$$

Again, symmetric rectangle window is used for frame weighing.

It will be more reasonable to consider the neighboring sub-clips when the similarity between two sub-clips is measured

$$S''_{ij} = \sum_{k=-m}^{m} w'_k S'_{i+k,j+k} \tag{8}$$

where w'_k is also set as equal weighting.

The transition probability from ith sub-clip to jth sub-clip is determined by $S''_{i+1,j}$. It can be calculated by a similar equation to (3).

IV. Unconstrained Synthesis Process

Once the transition probability between every two sub-clips has been found, the audio texture can be generated sub-clip by sub-clip. The issues here are, (1) to determine the order in which the sub-clips should be combined and played, and (2) to add varying effects into the building patterns. Since there is no boundary constraint in this case, the synthesis process is relatively free.

A. Determination of the Sequence Order

To select the sub-clip following sub-clip i, the simplest way is to select sub-clip j with the maximum probability P_{ij}. However, such a scheme always keeps the original sub-clip order, since $P_{i,i+1}$ always has the maximum probability [it equals to 1 according to (3)]. Moreover, in real applications, this scheme sometimes causes repetition of a small part of the original audio stream, especially at the end of the audio. This is because toward the end part of an audio/music clip, the sound lasts long and its energy gradually decreases, which makes the sub-clips near the end part very similar. Thus, the sub-clips at the end cannot find more similar succeeding sub-clips at the front part of the audio clip than those sub-clips around them. It causes these sub-clips repeated themselves, and thus the generated audio texture has jitters at the end and perceptually uncomfortable, as Fig. 5(a)

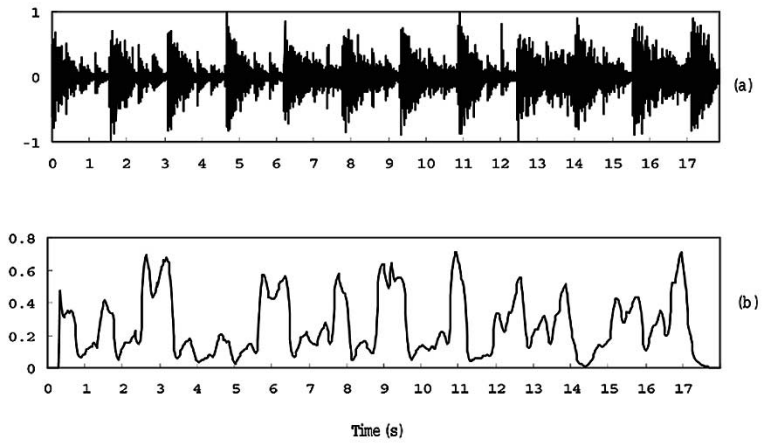

Fig. 4. Example of sub-clip and building pattern extraction from a music clip. (a) Digital audio data of a music clip and (b) corresponding novelty score curve.

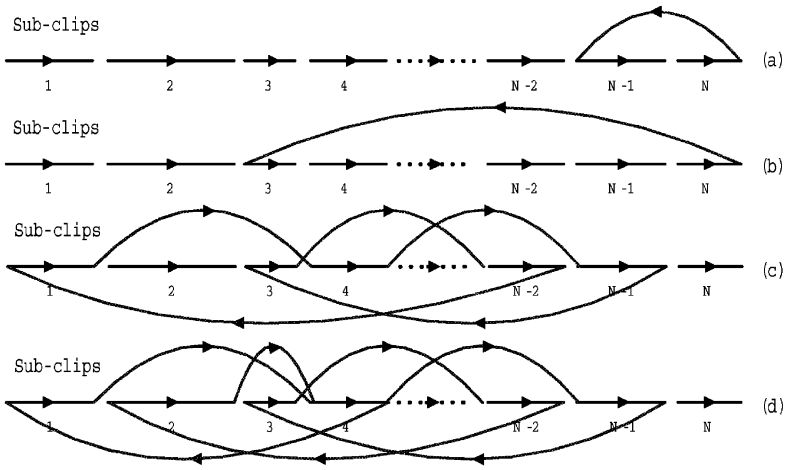

Fig. 5. Illustrations of sub-clip sequencing: each segment represents a sub-clip, the number under the segment represents the sub-clip index, and the arrowheads mean the order and connection between sub-clips. (a) a sub-clip sequence with jitters at the end; (b) a sub-clip sequence which removes jitters, but still keeps the original order with a big range repetition; (c) a sub-clip sequence which does not keep exactly the original order, but still repeats sub-clips with a fixed order; and (d) a sub-clip sequence with almost a random order.

illustrates. In Fig. 5, each segment represents a sub-clip, the number under the segment represents the sub-clip index, and the arrows mean the order and connection between sub-clips.

In order to solve the problem of above jitters at the end of an audio clip, we select sub-clip j with certain condition, as defined by the following equation:

$$j = \arg \max_{j \notin (i-r,i]} \{P_{ij}\}. \tag{9}$$

This means that the next sub-clip immediately following sub-clip i is searched in all sub-clips but not in its previous range in window $(i - r, i]$, where r is the size of the range. Larger value of r emphasizes a big jump while small value of r allows small jump. This condition forces the sub-clip at the end jumps to the front or middle part. An example result of sub-clip sequence after this constrain is shown in Fig. 5(b). It almost keeps the original order between sub-clips, and repeats in a large range.

In order to add variations to the original order between sub-clips, we can slightly modify the conditions in (9) as follows:

$$j = \arg \max_{j \notin (i-r,i+r)} \{P_{ij}\}. \tag{10}$$

It also prohibits from selecting the originally succeeding sub-clip, and hence disorders the original sub-clip sequence. An example of this kind of ordering result is shown in Fig. 5(c). From (10), it can be seen that the sub-clip order is actually fixed if parameter r is constant; and the sub-clip order is then repeated again and again.

To introduce more stochastic variation in the generated sub-clip sequence, we select a random one of the sub-clips in the following set as the subsequence of sub-clip i, with the constraints similar to (10)

$$j \in \{j | P_{ij} > P_0\} \bigcap j \notin (l - r, l + r) \tag{11}$$

where p_0 is a threshold and used to control the number of candidate sub-clips. Larger values of p_0 emphasize the very best transitions while smaller values of p_0 allow for greater variation at the cost of poorer transitions. It causes a sub-clip sequence with a completely random order, as Fig. 5(d) shows.

In real implementation, parameter r is randomly set for each selection of the subsequent sub-clip. It introduces a more general sub-clip sequencing results than a fixed r can. When $r = 0$, the original sub-clip order is probably kept.

Actually, in the sequence order determination, we are also planning to consider the amplitude smoothness and pitch continuity. It will be very helpful for perceptual ease of the final generated texture.

B. Adding Effects

Variations can be introduced to the sub-clips. In our implementation, potential variations include time scaling, pitching shifting, and amplitude setting. Different variations can be done by setting different values for the controlling parameters, which are all automatically and randomly set in our system. However, a parameter value for pitch-shifting should be applied to a group of consecutive frames (or a sub-clip) to avoid abrupt changes in pitch. A linear interpolation process is performed on the transitional frames between two groups in order to ensure that the pitch continuity.

An interpolation method or the TD-SOLA (Time Domain -Synchronous OverLap-Add) method [9], [10] is used for implementing time scaling and pitch shifting. Linear interpolation scales the audio waveform to an expected duration; however, it will change its time scale and pitch scale simultaneously. If one wants to change time scale and pitch scale independently, TD-SOLA can be simply employed. In the basic TD-SOLA system, time scaling and pitch shifting are performed directly on the audio waveform. The audio waveform is segmented into several windowed data, and then the compression or expansion is performed by re-arranging them on the time-axis. Some of the windowed data might be deleted or repeated in order to obtain the desired duration. Windowing and overlap-add techniques are used to fade smoothly between the segments.

C. Synthesis by Sequencing and Combining

Basically, an audio texture is generated as a sequence of sub-clips with possibly various effects, which are generated according to the method described in above section. However, there are many alternative ways of sequencing and combining the sub-clips to generate interesting audio textures. The two main methods are 1) sequence the sub-clips one by one without overlapping and 2) combine the sub-clips with some time-overlapping.

For example, given the sound of horse neighing, different effects are added to create new horse neighing by time-scaling and pitch shifting. Then, we can generate a sequence of neighing of a single horse by sequencing the different horse neighing effects end to end, which will be perceived as a horse neighing alone. We can also generate an effect in which different horse neighing effects are synchronously or asynchronously combined with some time-overlapping. It sounds like a group of horses neighing alternatively, as one falls, another rises. It is noted that, the method of combination with time-overlapping can generate perceptually better audio textures on an individual full audio clip than on a sub-clip, since sometimes a single sub-clip is not semantically meaningful. Moreover, combined with amplitude variation, more interesting audio effects can be generated. For instance, by applying certain variations in pitch and amplitude to a sequence of horse running, such as increasing amplitude and pitch firstly and then decreasing them at the end, we can generate an audio texture effect of horses running toward and then away from the listener.

In both sequencing and combining, TD-SOLA is used again to ensure the perceptual smoothness between two concatenated sub-clips.

Although all parameters can be randomly selected in our current system implementation, users can also choose parameter values based on their preferences, such as tendency of the pitch and amplitude in the sub-clip sequence, the method of sequencing or combining, the overlapping duration between different effects, etc. Manual adjusting can make the generated audio textures perceptually better.

V. Constrained Synthesis

In the constrained case, as Fig. 1 shows, the missing part of an audio clip will be restored or reconstructed by using audio textures. Unlike the unconstrained case, the synthesized part should be perceptually smooth at the joint points with the remaining audio clip. No uncomfortable break or click is expected at those points.

Before constrained synthesis, the missing part is detected at first based on the novelty curve. Fig. 6(a) shows an audio clip with a part missed and Fig. 6(b) shows the corresponding novelty score curve. Obviously, the missing part is the part that has nearly zero in novelty score curve. After the missing part is located, the similarity measure and the transition probability should be modified correspondingly, since the data of the missing part should not be used in the measurement. The modification is quite easy. We just need to modify the upper or low bound of the formula in (2) to ensure no data is used in the missing part.

In such constrained case, frame instead of sub-clip is used as the synthesis unit, due to the following reasons.

1) The missing part usually does not begin at the beginning of one sub-clip and end at the ending of another sub-clip. It may begin or end at the inner of the sub-clips, just as Figs. 1 and 6 shows.
2) It is difficult to estimate how many sub-clips should be inserted into the missing part, since each sub-clip is not of equal length.
3) It is possible that the length of the missing part of an audio clip is less than that of a sub-clip.

It should also be noted that, in this case, it is impossible to restore the audio clip exactly the same as the original one, since we don't know what it originally was. Thus, our objective is to generate an audio clip, which can replace the missing part and can be smoothly integrated into the remaining audio without uncomfortable perception. It is feasible based on the characteristics of self-similarity of the remaining audio texture. Thus, the key issue is how to determine the frame sequence which can fill in to replace the missing part in the audio clip.

In the following sub-sections, we will introduce our approaches to frame sequence determination for audio texture restoration. We assume that the missing region of an audio clip, or the region to be synthesized, is from frame i_0 to frame j_0, as Fig. 6 shows.

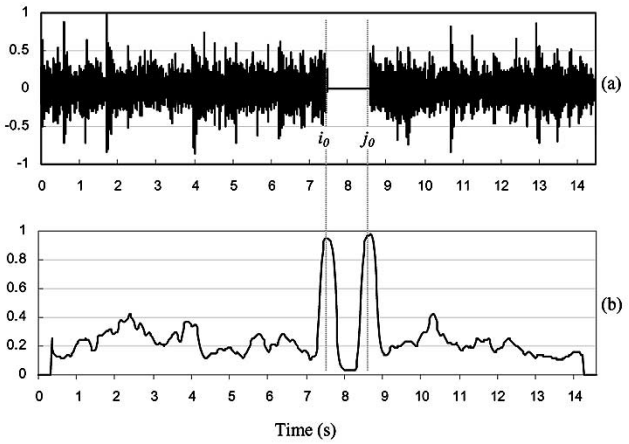

Fig. 6. Example of the missing part detection. (a) Waveform of an audio clip with a long duration of data is lost and (b) novelty scores for detecting which part is missed in the audio clip.

A. Frame Sequence Determination

Since the missing part is similar to the remaining part, the simplest method that can be used for constrained audio texture synthesis is to replace the missing part by using the most similar piece of the same length in the remaining audio clip. In this case, the neighboring frames around are used to represent the feature of the missing part. This method is feasible but may result in some problems, such as the following.

1) This method considers the similarity of the context of the missing part but ignores the perception continuity at the joint point i_0 and j_0, where possible abrupt changes or clicks may occur.
2) Since this method just copies a segment of remaining audio to replace the missing part, such exact repeat of another part in a very close time may cause discomfort to some listeners.

In order to solve these problems, another optimization method is used in our real implementation, where, we restore the missing part frame by frame. The main problem is to determine the frame sequence which can be smoothly inserted and replace the missing part without uncomfortable perception. The optimal frame sequence should satisfy

1) maximize the transition probability from frame i_0 to frame j_0;
2) keep perceptual smoothness.

The determination of frame sequence from i_0+1 to j_0-1 can be described in the following mathematic model:

$$\max P(i_0 \rightarrow j_0) = P_{i_0,i_0+1} \cdot P_{i_0+1,i_0+2} \cdots P_{j_0-2.j_0-1} \cdot P_{j_0-1,j_0} \tag{12}$$

with the constraints

$$P_{i,i+1} > p_0, i_o \leq i \leq j_0 - 1 \tag{13}$$

where p_0 is a threshold to select a frame with a sufficiently large transition probability can be used to control the perceptual smoothness.

However, the feasible solution space of this problem is very large. Suppose under the constrained condition in (15), the number of potential candidate frames after a given frame is N; and the number of missing frames is $M = j_0 - i_0 - 1$, the size of the feasible solution space is about N^M. This is too large for an exhaustive search to find the optimal solution when N and M are large. Thus, dynamic programming is utilized to find a sub-optimal solution, with an *O(NM)* complexity.

In general, dynamic programming is used to find a path from i_0 to j_0 which has the minimum cost. Hence, some modifications are needed to make the problem suitable for dynamic programming. Therefore, the transition probability P_{ij} should be changed to the cost of moving from point i to point j, which can be defined as

$$c_{ij} = \begin{cases} -InP_{ij} & (P_{ij} > P_0) \\ \infty & (P_{ij} < P_0). \end{cases} \tag{14}$$

After determining the frame sequence, the TD-SOLA method is used again to combine and smooth each two concatenated frames.

B. Global Amplitude Trend Consideration

In the above algorithm, we only considered the characteristics of spectral similarity. However, the optimal frame sequence had better keep not only the spectral characteristics, but also energy characteristics. The global amplitude variation trend in the missing part is estimated based on the most similar piece of the same length in the remaining audio clip. In this case, the neighboring frames around are used to represent the feature of the $[i_0, j_0]$ region, since the frames between $[i_0, j_0]$ are unknown. Suppose the previous m frames before i_0 and the next m frames following j_0 are considered to represent the amplitude characteristics of the context of $[i_0, j_0]$, the feature vector of $[i_0, j_0]$ can be represented as

$$\bar{A}_{i_0,j_0} = (A_{i_0-m}, A_{i_0-m+1}, \ldots A_{i_0-1}, A_{j_0+1}, A_{j_0+2}, \ldots A_{j_0+m}). \tag{15}$$

Then, the similarity between the context around $[i_0, j_0]$ and the context around $[i', j']$ are calculated from correlation

$$S_g\left(\bar{A}_{i_0,j_0}, \bar{A}_{i',j'}\right) = \frac{\bar{A}_{i_0,j_0} \bullet \bar{A}_{i',j'}}{|\bar{A}_{i_0,j_0}| \bullet |\bar{A}_{i',j'}|} \tag{16}$$

where $j' - i' = j_0 - i_0$.

After computing the similarity between the context of $[i', j']$ and the context of each $[i', j']$ with the same length, the most similar part $[i^*, j^*]$ determined by the following equation is used to estimate the global amplitude tendency of the missing part:

$$[i^*, j^*] = \arg\max\{S_g(\bar{A}_{i_0,j_0}, \bar{A}_{i',j'})\}. \tag{17}$$

Thus, when to determine the optimal frame sequence, the similarity of each two frames should be composed of two parts: spectral similarity and global amplitude tendency similarity. That is

$$S = \lambda S_1 + (1-\lambda) S_2 \tag{18}$$

where S_1 and S_2 are the spectral similarity between two frames and their amplitude tendency similarity respectively, λ is the corresponding weight. In general, $\lambda > 0.5$, since spectral similarity are more important and the amplitude estimation is

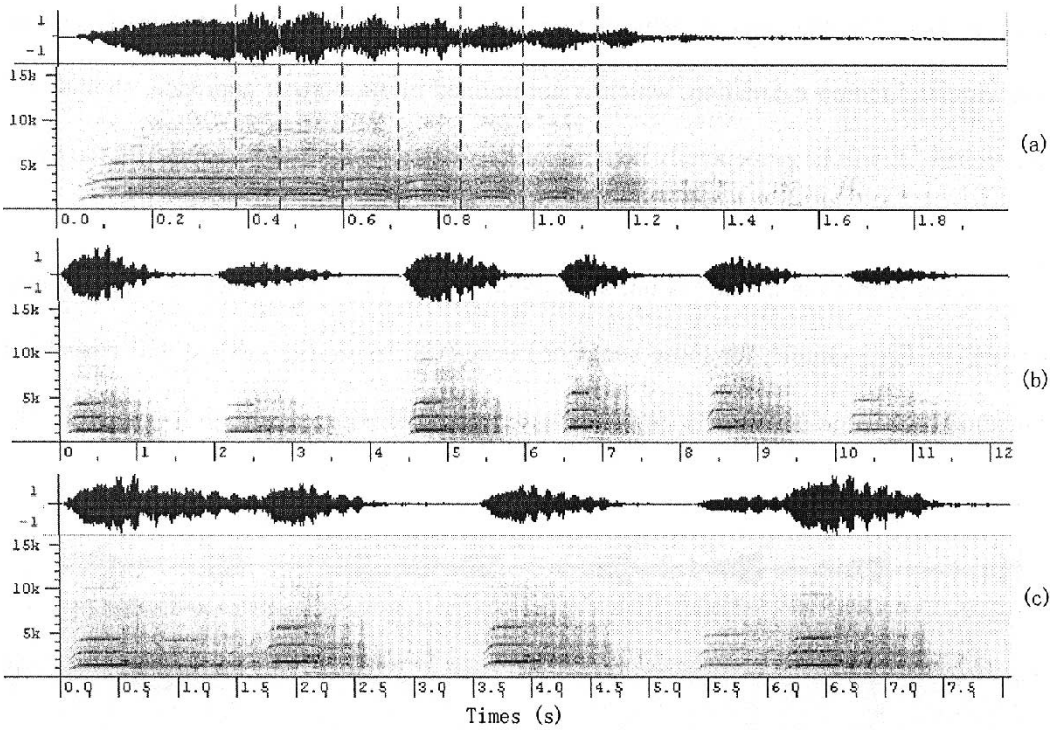

Fig. 7. Different audio texture effects and its original sound of horse neighing: (a) waveform and spectrogram of the original sound; (b) waveform and spectrogram of an audio texture generated by sequencing different variation effects; and (c) waveform and spectrogram of an audio texture generated by combining variation effects with some time-overlapping.

not very precise. The transition probability between each two frames can then be calculated correspondingly, based on the (3).

VI. Applications and Preliminary Evaluations

In Section VI, some applications on constrained or unconstrained audio texture synthesis are presented. Some preliminary examples are presented on our website at http://research.microsoft.com/~llu/AudioTextures/. The readers who are interested in this work can access to them and compare the original sounds and the synthesized or restored audio textures.

A. Unconstrained Audio Texture Synthesis and its Applications

Unconstrained audio texture synthesis can be used in many applications, such as background sound and audio effects. We have generated some audio effects using the audio textures idea for evaluation, which include horse neighing, rooster crowing, thunder, explosion, raining, stream, ripple, and simple music clips. All the original audio clips are 2-15 s long, sampled at the rate of 8 KHz or 32 KHz, mono channel, and encoded by 16 bit per sample. Some generated texture examples are shown in the following.

Horse neighing. The input audio is about 2 s long with 32 KHz sampling rate. Its waveform and spectrogram is illustrated in Fig. 7(a). It contains just one neigh of a single horse and can be taken as one sub-clip. In order to introduce more flexibility, eight sub-clips are extracted based on the amplitude contour, such as Fig. 7(a) shows. These sub-clips correspond to the start, the end and several vibrations in the middle of a neigh. Certain variation effects are added for each sub-clip by adjusting the parameters of duration, pitch, and amplitude. These parameters are variation ratios corresponding to the original value and are set randomly in a certain range. Different neighing of horse is generated by randomly adjusting these parameters.

Based on these different effects, two textures are synthesized. The first one is created by connecting the different neighs as a temporal sequence, generating a sound that a horse is neighing continuously. Fig. 7(b) shows the corresponding waveform and spectrogram. From the figure, we can see that each effect is a different variation of the original clip, by pitch shifting, time scaling, and amplitude changing. Another one is created by combining different neighs with some time-overlapping, as shown in Fig. 7(c). The result is a sound of a group of horses that are neighing in turns, synchronously or asynchronously.

Although all parameters are randomly selected in our current prototype system, users can also adjust those parameters by themselves. Manual adjusting can make the generated audio textures perceptually better, especially the adjusting on the overlapping duration between different effects in combination. Some aesthetic sense is usually necessary in deciding which effects should be overlapped and how they are overlapped, in order to create a realistic sound effect.

Simple music. This example shows how this algorithm works on simple music, since music is always more complex than other audio types. This given music clip is about 12 s long with 8 KHz sampling rate, as illustrated in Fig. 8(a). This clip has a simple rhythm with some percussion sounds indicating beats. Dozens of building patterns are extracted, without any variation effect

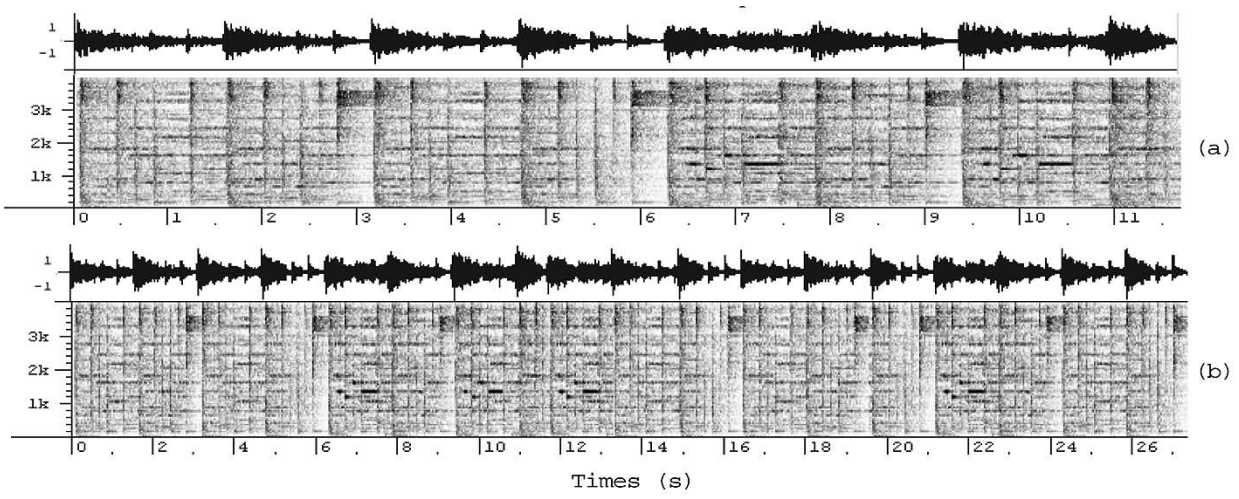

Fig. 8. Audio texture generation of a music clip: (a) waveform and spectrogram of the original music clip sound and (b) waveform and spectrogram of the generated music texture with the duration of about 27 s.

is added on them in this case. The final audio texture, with duration of about 27 s, is synthesized by sequencing the sub-clips based on their transition probability. The corresponding waveform and spectrogram are shown in Fig. 8(b). We can see that the generated audio texture keeps the structure of the original music clip well.

One of the problems with the method is potential change of the rhythm. In the original sound, the duration between contiguous beats is uniformly about 1.56 s. However, in the generated texture, there is a tempo variation at about the 12th s: the duration between two beats becomes 0.78 s. It can be clearly seen from Fig. 8(b). Beat tracking and tempo estimation, which is not utilized in our current approach, should be helpful to solve such a problem. It will be perceptually better to align the percussion/drum sound with the estimated beat position.

Another problem is that the method is not so suitable for complex songs such as arbitrary music pieces from compact disks. For example, for those songs or music with lyrics, the method will probably fail. It is because the method could not discriminate the meaning of lyrics and organize the context accordingly.

Stream. This example is used to show how to generate audio textures using individual frames when no obvious building pattern is found. The input audio is a sound of stream of about 11 s long. It does not have obvious pitch information and is more like a noise signal. Thus, it can be used to synthesize audio textures more freely.

In the generation process, it is firstly divided into 25 ms-long frames with 12.5 ms overlapping. Then, an audio texture of random duration is generated by sequencing these frames instead of sub-clips. Variations on time-scaling and pitch-shifting are set for each 1-s segment, to generate the effects of different stream speed and loudness. In order to keep the perceptual smoothness of the generated texture, the parameters are linearly interpolated between the adjacent segments to prevent them changing too dramatically. The finally generated texture is a stream of infinite length, with some variations in stream speed and amplitude. Corresponding waveform and spectrogram is not given for this simple example.

B. Constrained Audio Texture Synthesis and Audio Texture Restoration

Constrained audio texture synthesis can be used in many audio restoration applications, such as the error concealment for audio/music delivery with packet loss on the Internet. Compared with most of the traditional error concealment methods, which only dealt with errors with a short length (typically around 20 ms) [11], our method can restore the audio with loss of a much longer length, such as one second.

The proposed method does not attempt to estimate the original missing data, but to generate a replacement signal which is similar in the structure and can be smoothly integrated into the remaining data. It is difficult to give an objective evaluation, since it cannot evaluate performance simply by comparing the waveform between the original audio and the restored audio. Some casual subjective evaluations by some subjects have indicated that the proposed method gives promising results

Some examples of audio texture restoration are implemented by using the constrained audio texture synthesis algorithm presented above, which are also put in the website, comparing with the original audio and the damaged audio clip. The original audio clips are all about 10–30 s long, sampled at the rate of 16 KHz or 32 KHz, mono channel, and encoded by 16 bit per sample.

An example of our constrained audio texture synthesis algorithm is illustrated in Fig. 9. Fig. 9(a) shows the waveform and spectrogram of an original audio clip, which is an excerpt of a music clip with a slow tempo and simple rhythm. Fig. 9(b) shows the audio clip which is derived from (a) but loses some data of a long duration (from 17.8 s to 19.6 s). The restored audio clip by using the proposed frame sequence determination algorithm presented in Section V is illustrated in Fig. 9(c). From the figure, we can see that the restored audio keeps the pitch continuity well, although its pitch information is a little different from the original pitch curve. We can also see that the restored amplitude curve is similar to that of the missing part. It means that the amplitude tendency estimation in our algorithm also works well.

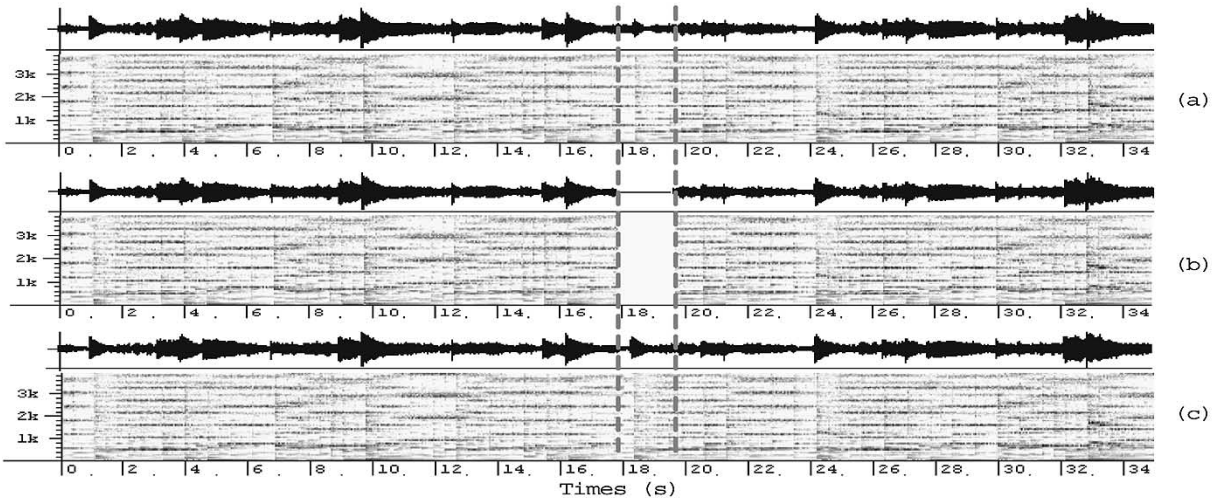

Fig. 9. Example of the audio texture restoration process. (a) waveform and spectrogram of the original audio clip; (b) a long duration (17.8 s to 19.6 s) of data is lost in the original audio clip; and (c) the restored audio clip by our constrained audio texture synthesis method. As can be seen clearly, it keeps pitch continuity and amplitude tendency very well.

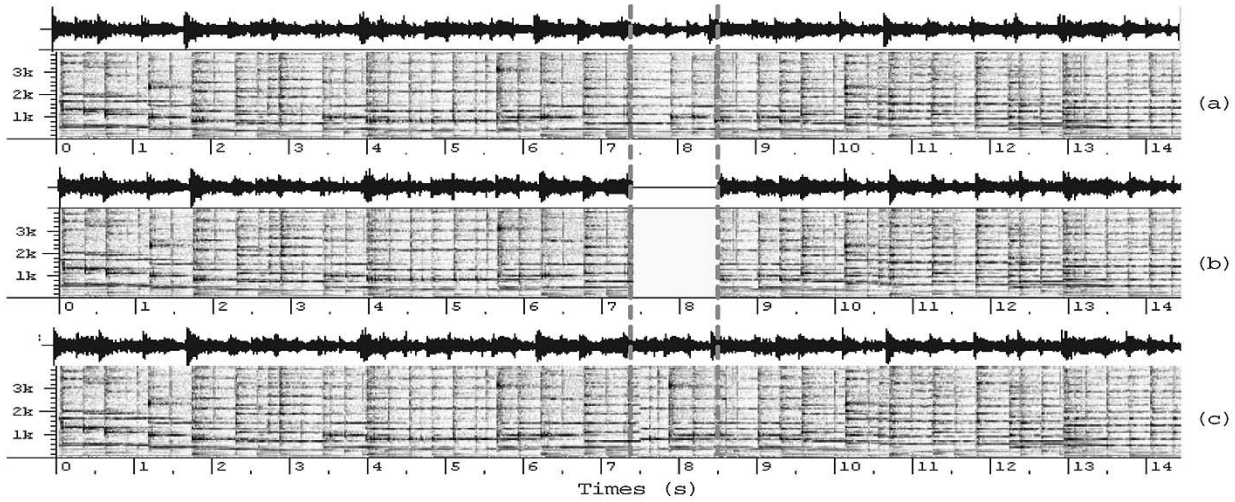

Fig. 10. Example of audio texture restoration with artifacts. (a) waveform and spectrogram of the original audio clip; (b) a long duration (7.4 s–8.5 s) of data is lost in the original audio clip; and (c) the restored audio clip has a pitch discontinuity between 7.4 s–7.9 s.

Actually, in some experiments of audio textures restoration, pitch continuity is not always well reserved. This is because that the selected feature in our approach, MFCC, deemphasizes the pitch information, which is very important in such case. One of such example is illustrated in Fig. 10. Comparing Fig. 10(a) with Fig. 10(c), it can be seen that the pitch continuity in the restored audio is lost, especially in the duration from 7.4 s to 7.9 s. Thus, some artifacts are produced and perceptual smoothness is decreased. To solve such a problem, pitch and harmonics information should be considered, as did in music mosaicing [18] and sound synthesis based on sinusoids plus noise model [20].

As mentioned before, another potential problem may also exist with rhythm. Beat tracking and tempo estimation are also helpful in constrained audio texture synthesis.

C. Subjective Evaluations

To provide more comprehensive understanding of the effect of our approach, a simple subjective evaluation is performed. In this experiment, 15 subjects are asked to score the generated or restored audio textures. Subjects were not given any special instructions on how to measure the performance. It is solely based on their subjective perceptions.

In the subjective evaluation on unconstrained audio texture synthesis, two criteria are used: *smoothness* and *variety*. *Smoothness* is used to measure if the generated texture is natural or smooth, while *variety* is used to evaluate if the generated texture is monotonic. The rating level is 3, 2, and 1, which represents satisfying, acceptable and bad, respectively. The detail results are illustrated in the Fig. 11. It can be seen that most of the subjects think the generated textures are smooth and not monotonic.

In the evaluation on constrained audio texture restoration, we compared our approach with other two restoration methods. One method is from traditional approach to error concealment. Wang [11] reported that the prior works for error concealment of small segments include: 1) muting, 2) repeating prior packet, 3) interpolation, and 4) time-scale modification. Since the first, the third

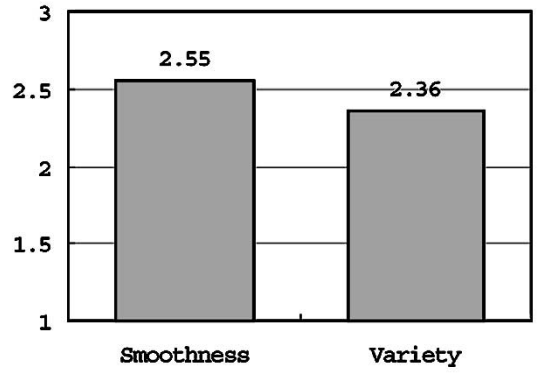

Fig. 11. Subjective evaluation on unconstrained audio synthesis.

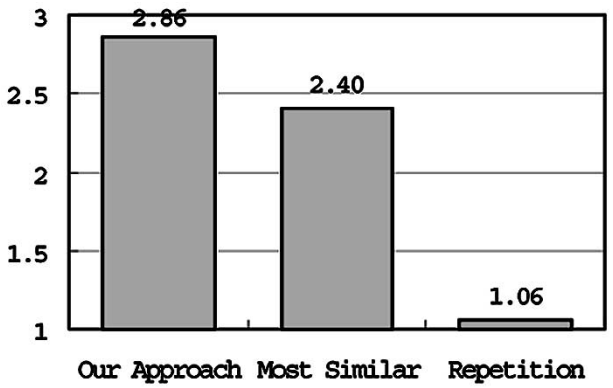

Fig. 12. Subjective comparison among our method, filling the most similar segment and the repetition of the previous frame.

and the last method are not suitable for the restoration of long segment in our case, only the second method is use to compare with our method. Another method for comparison is to replace the missing part by using the most similar piece of the same length in the remaining audio clip, which is mentioned in the Section V-A. The detail comparison results are illustrated in the Fig. 12.

From the Fig. 12, it can be seen that our approach is much better than the other methods. Most of the subjects are satisfactory with our restoration; while the 'frame repetition' method almost got a 'bad' score, since just repetition of a frame in a long segment always causes monotonic and perceptual uncomfortable. The method of using the most similar segment is also acceptable by most of subjects. However, since it has possible abrupt changes at the boundaries, it is not as satisfying as our method.

D. Limitations

Although the proposed algorithms work promisingly in our testing data, it still has some limitations. This sub-section concludes the limitations to help understand the algorithm better, although most of them have already been discussed in Sections VI-A–C.

1) The proposed algorithms are more suitable for those audio clips/textures with simple structures. For example, in constrained audio texture restoration, the missing signal is similar to the remaining part for audio clips of simple structures. However, for those audio clips of complex structures, it is difficult to restore a perceptually smooth signal. It works similarly in the unconstrained case. Moreover, for those songs or music with lyrics, the proposed approach usually fails.
2) The proposed algorithm uses MFCC as the only spectral feature. However, MFCC deemphasizes pitch information which might be important for certain types of sounds. It may introduce some potential problems of pitch discontinuity.
3) Another problem with the method is the potential change of rhythm. Since we did not keep the beat and tempo information in the algorithm, the beat position might be misplaced in the generated textures.
4) There is no perception criterion to control the texture synthesis procedure. Local perception and global perception are both extremely useful in the proposed approach, but they are difficult to obtain.

VII. Conclusion

In this paper, we have introduced a new audio media, called audio textures. An audio texture is an audio sequence of arbitrary length generated from a short clip of audio example. It consists of consecutively connected patterns that are perceptually similar to those contained in the example clip but present variations. A method has been proposed for audio texture construction, which includes extraction of the basic patterns from the original clip, making of variations of the basic patterns, and connection of the variable patterns into a long sequence.

We have also extended the idea to constrained audio texture synthesis, where a part of audio signal is lost and audio texture is used to restore the missing part. The restored part can be smoothly integrated into the original audio clip without any perceptually discomfort. In order to do that, we have been proposed methods for detection of the missing part and determination of the filling frame sequence.

There are many potential applications for audio textures such as lullabies, game music, background sounds and other effects. Another potential application is that it is a good choice for audio compression. It can also be used in audio signal restoration, such as error concealment for audio/music delivery with packet loss on the Internet. Some applications have been presented in the paper for both the unconstrained and constrained audio texture synthesis. Casual subjective evaluations indicated that the proposed methods are encouraging.

Audio texture synthesis is a new concept. We also hope the new concept could inspire more research work in the audio and related fields, such as computer music, audio content analysis, audio retrieval, and audio compression.

To solve the limitations of current algorithm, the audio texture synthesis technique should be improved in several aspects in the future work. In the analysis step, we currently just used a correlation to measure the similarity between each pair of frames. It will be more useful if we could find a perceptual similarity measurement. In the synthesis step, we generated frame sequences or sub-clip sequences based on local similarity only. How to control the global perception of the generated texture is still a difficult task. Other features, such as harmonics and pitch, will

also be helpful for audio texture synthesis to keep the perceptual smoothness. Beat tracking and tempo estimation are also worth considering in the algorithm to keep a certain rhythm. In evaluations, it will be better if more effective evaluation could be used on our algorithm. We would also extend our work to more traditional music clips. Thus, more powerful signal processing methods are also needed.

ACKNOWLEDGMENT

The authors should acknowledge S. Z. Li's help in developing and discussing the original idea of Audio Textures. The authors also thank Y. Mao from Zhejiang University for implementing some part of the algorithms.

REFERENCES

[1] A. Schodl, R. Szeliski, D. H. Salesin, and I. Essa, "Video textures," in *Proc. SIGGRAPH 2000*, July 2000, pp. 33–42.
[2] J. S. de Bonet, "Multi-resolution sampling procedure for analysis and synthesis of texture images," in *Proc. SIGGRAPH'97.*, 1997, pp. 361–368.
[3] A. A. Efros and T. K. Leung, "Texture synthesis by nonparametric sampling," in *Proc. IEEE Int. Conf. Computer Vision*, 1999.
[4] J. Foote, "Visualizing music and audio using self-similarity," in *Proc. ACM Multimedia '99*, Orlando, Florida, November 1999, pp. 77–80.
[5] L. Lu, H. Jiang, and H. J. Zhang, "A robust audio classification and segmentation method," in *Proc. 9th ACM Multimedia*, 2001, pp. 203–211.
[6] S. Z. Li, "Content-based classification and retrieval of audio using the nearest feature line method," *IEEE Trans. Speech Audio Processing*, vol. 8, pp. 619–625, Sept. 2000.
[7] J. Foote, "Automatic audio segmentation using a measure of audio novelty," in *Proc. Int. Conf. Multimedia and Expo (ICME)*, 2000.
[8] L. Lu, S. Li, L. Wenyin, H. J. Zhang, and Y. Mao, "Audio textures," in *Proc. ICASSP2002*, vol. II, 2002, pp. 1761–1764.
[9] E. Moulines and F. Charpentier, "Pitch-Synchronous waveform processing techniques for text-to-speech synthesis using diphones," *Speech Commun.*, vol. 9, pp. 453–467, 1990.
[10] H. Valbret, E. Moulines, and J. P. Tubach, "Voice transformation using PSOLA technique," in *Proc. ICASSP-92.*, 1992.
[11] Y. Wang, "A beat-pattern based error concealment scheme for music delivery with burst packet loss," in *Proc. Int. Conf. Multimedia and Expo (ICME'01)*, 2001, pp. 73–76.
[12] A. Stenger, K. B. Younes, R. Reng, and B. Girod, "A new error concealment technique for audio transmission with packet loss," in *Proc. Eur. Signal Processing Conf. (EUSIPCO 96)*, Trieste, Italy, Sept. 1996, pp. 1965–1968.
[13] S. J. Godsill, P. J. W. Rayner, and O. Capp'e, "Digital audio restoration," in *Applications of Digital Signal Processing to Audio and Acoustics*, K. Brandenburg and M. Kahrs, Eds. Norwell, MA: Kluwer, 1996.
[14] L. Rabiner and B. H. Juang, *Fundamentals of Speech Recognition*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[15] K. D. Martin, "Sound-Source Recognition: A Theory and Computational Model," Ph.D. dissertation, Massachusetts Inst. Technol., Cambridge, 1999.
[16] M. Davy and S. Godsill, "Detection of abrupt spectral changes using support vector machines an application to audio signal segmentation," *Proc. ICASSP'02*, vol. II, pp. 1313–1316, 2002.
[17] L. Lu, S. H. Li, and J. Zhang, "Content-based audio segmentation using support vector machines," in *Proc. ICME'01*, Tokyo, Japan, 2001, pp. 956–959.
[18] A. Zils and F. Pachet, "Musical mosaicing," in *Proc. Cost G-6 Conf. Digital Audio Effects DAFX-01*, Limerick, Ireland, 2001.
[19] C. Roads, "Asynchronous granular synthesis," in *Representations of Musical Signals*, G. De Poli, A. Piccialli, and C. Roads, Eds. Cambridge, MA: MIT Press, 1991, pp. 143–185.
[20] X. Serra and J. Smith, "Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition," *Comput. Music J.*, vol. 14, no. 4, pp. 12–24, 1990.
[21] E. Klabbers and R. Veldhuis, "Reducing audible spectral discontinuities," *IEEE Trans. Speech Audio Processing*, vol. 9, pp. 39–51, Jan. 2001.

Lie Lu received the B.S. and M.S. degrees from Shanghai Jiao Tong University, China, both in electrical engineering, in 1997 and 2000, respectively.

From 2000, he is with Microsoft Research, Asia, where he is currently an Associate Researcher in the Media Computing Group. His current interests are in the areas of pattern recognition, content-based audio analysis, and music analysis. He has authored more than 20 publications and has nine patents pending in these areas.

Liu Wenyin (M'99–SM'02) received the B.Eng. and M.Eng. degrees in computer science from the Tsinghua University, Beijing, China, in 1988 and 1992, respectively, and the D.Sc. degree in information management engineering from The Technion—Israel Institute of Technology, Haifa, in 1998.

He had worked at Tsinghua as a faculty member for three years and at Microsoft Research China/Asia as a full time Researcher for another three years. Currently, he is an Assistant Professor in the Department of Computer Science at the City University of Hong Kong. His research interests include graphics recognition, pattern recognition and performance evaluation, user modeling and personalization, multimedia information retrieval, personalization information management, object-process methodology, and software engineering and Web services engineering. He has authored more than 100 publications and seven patents pending in these areas.

Dr. Wenyin played a major role in developing the Machine Drawing Understanding System (MDUS), which won First Place in the Dashed Line Recognition Contest (http://www.loria.fr~tombre/grec95.html) held during the First IAPR Workshop on Graphics Recognition at Pennsylvania State University in 1995. In 1997, he won Third Prize in the ACM/IBM First International Java Programming Contest (ACM Quest for Java'97) (http://www.acm.org/jquest/webquest1.html). In 2003, he was awarded the ICDAR Outstanding Young Researcher Award by the International Association for Pattern Recognition (IAPR) for his significant impact in the research domain of graphics recognition, engineering drawings recognition, and performance evaluation. He co-chaired the Fourth International Contest on Arc Segmentation held during the Fourth IAPR Workshop on Graphics Recognition, Kingston, ON, Canada, in September 2001, and chaired the Fifth International Contest on Arc Segmentation held during the Fifth IAPR Workshop on Graphics Recognition, Barcelona, Spain, in July 2003. He is currently organizing the Sixth IAPR Workshop on Graphics Recognition to be held in Hong Kong in August 2005. He is a member of the IEEE Computer Society, the IEEE Education Society, and ACM.

Hong-Jiang Zhang (S'90–M'91–SM'97–F'04) received the Ph.D. from the Technical University of Denmark and the B.S. from Zhengzhou University, China, both in electrical engineering, in 1982 and 1991, respectively.

From 1992 to 1995, he was with the Institute of Systems Science, National University of Singapore, where he led several projects in video and image content analysis and retrieval and computer vision. He also worked at MIT Media Lab, Cambridge, MA, in 1994 as a Visiting Researcher. From 1995 to 1999, he was a Research Manager at Hewlett-Packard Labs, where he was responsible for research and technology transfers in the areas of multimedia management, intelligent image processing, and Internet media. In 1999, he joined Microsoft Research Asia, where he is currently a Senior Researcher and Assistant Managing Director in charge of media computing and information processing research. He has authored three books, over 200 refereed papers and book chapters, seven special issues of international journals on image and video processing, content-based media retrieval, and computer vision, as well as numerous patents or pending applications.

Dr. Zhang is a member of ACM. He currently serves on the editorial boards of five IEEE/ACM journals and a dozen committees of international conferences.