Audio Textures

Georg Holzmann

Feb. 2006

Audio Texture ist eine Methode, aus einem gegebenen kurzen Audiobeispiel einen beliebig langen Audiostream zu generieren.
Das Audiobeispiel wird dabei zuerst analysiert und anschließend anhand von spektralen Eigenschaften (MFCC) in kleinere Teile unterteilt. Letzlich wird nun aus diesen ein beliebig langer Audiostream generiert.
Diese Methode kann ebenfalls verwendet werden, um längere Drop-Outs in einem Audiostream auszubessern bzw. aufzufüllen (Audio Texture Restauration).

Inhaltsverzeichnis

1 Einführung

1.1 Was sind Audio Textures

Audio Textures, nach [4], sind eine Methode um aus einem kurzen Audiosample einen beliebig langen Audiostream zu generieren, der dem kurzen Beispielsample möglichst ähnlich ist, aber trotzdem nicht monoton bzw. unnatürlich klingt.
Um dies zu erreichen wird dieses Audiosample analysiert und in Basic Building Patterns eingeteilt, mit denen anschließend der Audiostream synthetisiert wird.
Mögliche Anwendungsgebiete sind Game Music, Background Music, Screen Saver Sounds, ...
Der Vorteil dabei ist, dass wenig Speicherplatz für langen Klang verwendet werden muss. Außerdem können Audio Textures zur Audio Restauration verwendet werden, um längere Drop-Outs in Daten aufzufüllen (z.B. Internet-Streaming etc.).

Verwandte Verfahren sind z.B. Video Textures oder Musical Mosaicing.
Bei Video Textures (siehe [2]) wird aus einem kurzen Videoclip ein beliebig langes Videosignal generiert, indem die Abspielreihenfolge der einzelnen Frames (unmerkbar) verändert wird - z.B. Fische die in einem Aquarium schwimmen, bewegte Flamme einer Kerze, Flagge die sich im Wind bewegt ...

Hingegen werden beim Musical Mosaicing (siehe [1]) Sampleausschnitte auf Grund von angegebenen Eigenschaften ausgesucht und zu neuen Sequenzen (= Mosaik) zusammengesetzt. Es benötigt also, im Gegensatz zu Audio Textures, eine gesamte Sample-Datenbank.

Im folgenden Text wird zwischen zwei Arten von Audio Textures unterschieden: Unconstrained und Constrained Synthesis.
Unconstrained Synthesis bedeutet, dass eine beliebig lange Audiosequenz generiert wird, welche ähnlich dem Ausgansmaterial ist, aber ständig variiert.
Von Constrained Synthesis, oder Audio Texture Restoration, spricht man, wenn ein größerer Teil eines Audiosignals verloren ist und wiederhergestellt werden soll. Dieser Syntheseprozess ist nun von Anfangs- und Endpunkt des Fehlers begrenzt, daher constrained.

1.2 Systemüberblick

Das Gesamtsystem wird sinnigerweise in 2 Teilabschnitte eingeteilt: Analyse und Synthese (siehe Abbildung 1).

1.2.1 Analyseprozess

Bei der Constrained Synthesis wird zuerst Anfangs- und Endpunkt des Fehlers lokalisiert.
Zur Feature Extraction werden die Mel-Frequency Cepstral Coefficients (MFCCs) verwendet (siehe Abschnitt 2.1), um die Struktur des Klanges zu analysieren.
Anhand dessen wird der Audio Clip nun in Basic Building Patterns (= Sub-Clips) unterteilt, wobei ein Sub-Clip aus einem oder mehreren Frames bestehen kann.
Zuletzt werden noch Ähnlichkeit und Übergangswahrscheinlichkeit zwischen allen Sub-Clips berechnet.

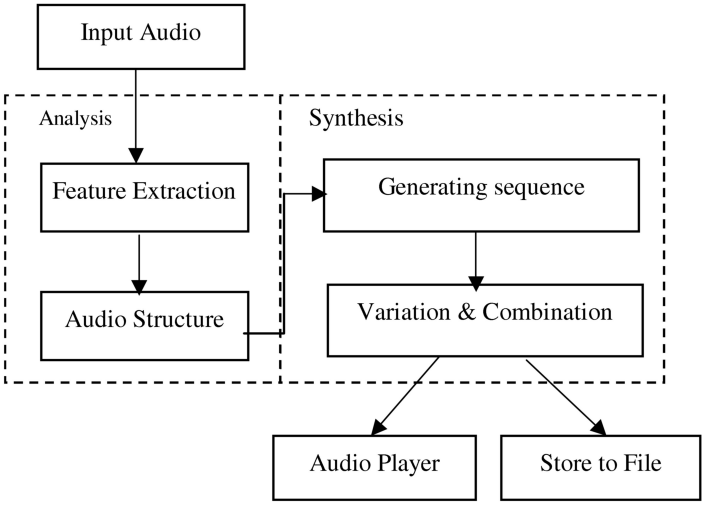

Abbildung 1: Diagramm des Systems (Quelle: [3])

1.2.2 Syntheseprozess

Zuerst wird eine Sub-Clip Sequenz anhand der berechneten Übergangswahrscheinlichkeiten generiert (bei Constrained Synthesis muss Übergang an den Eckpunkten beachtet werden).
Anschließend können verschiedene Effekte beigefügt werden: unterschiedliche Sub-Clip Sequenzen; Time Scaling und Pitch Shifting (implementiert mit Synchronous Overlap-Add (SOLA) Methode), ...

2 Analyseprozess

Im Analyseprozess wird der Audio-Clip zuerst in Blöcke engeteilt (Frames), aus denen die MFCC Koeffizienten berechnet werden.
Anhand dieser wird Similarity (Ähnlichkeit) and Transition Probability (Übergangswahrscheinlichkeit) zwischen den einzelnen Frames berechnet und anschließend der Audio-Clip in Sub-Clips (= Basic Building Blocks) unterteilt, wobei ein Sub-Clip aus einem oder mehreren Frames bestehen kann.

2.1 MFCC-Einführung

MFCC bedeutet Mel Frequency Cepstral Coefficients (siehe [5]), das sind wahrnehmungsangepasste, spektrale Eigenschaften, mit denen man leicht Signale auf Ähnlichkeit überprüfen kann (Verwendung in z.B. speech recognition, ...). Ich möchte nun kurz das Prinzip der MFCC vorstellen (siehe Abbildung 2):

1. Das Signal wird in Blöcke (= Frames) geteilt und gefenstert.
2. Transformation in den Frequenzbereich mit Hilfe der DFT.

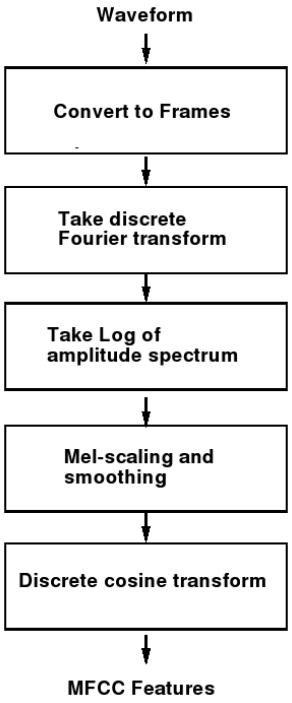

Abbildung 2: Diagramm zur Generierung der MFCC Feature-Vektoren (Quelle: [5])

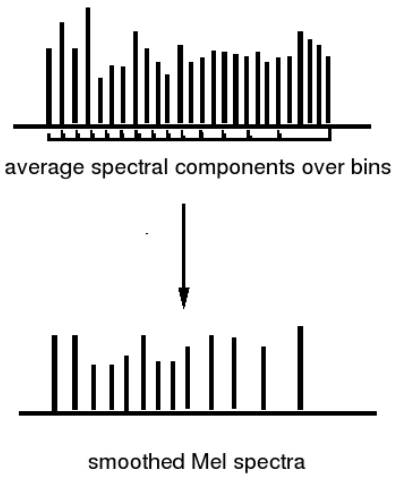

Abbildung 3: Mel-Scaling and Smoothing, Durchschnitt über Frequenzbins (Quelle: [5])

3. Nun wird der Logarithmus des Amplitudenspektrums genommen (Phaseninformationen werden vernachlässigt), da die Wahrnehmung der Lautheit auch eher logarithmisch funktioniert.

4. Sog. *Mel-Scaling and Smoothing*, was bedeutet, dass aus z.B. 256 Bins 40 gemacht werden, mit Hilfe der Berechnung des Durchschnitts von mehrere benachbarten Bins.
   Diese Berechnung ist wahrnehmungsangepasst und wird anhand der Mel-Skala durchgeführt, sodass z.B. tiefe Frequenzen betont werden etc. (siehe Abbildung 3)

5. Diese übriggebliebenen 40 Werte sind nun noch hoch korreliert. Deshalb wird noch eine Transformation angewendet, um eine minimale Anzahl unkorrelierter Werte zu bekommen.
   Theoretisch würde die Karhunen-Loeve (KL) Transformation dieses erreichen (oder equivalent dazu Principal Component Analysis (PCA)), diese kann jedoch durch eine Diskrete Cosinus Transformation angenähert werden, um den Rechenaufwand zu reduzieren.
   Dadurch können die 40 korrelierten Werte auf z.B. 13 unkorrelierte Werte pro Frame reduziert werden.

2.2 Similarity Measure und Transition Probability

Nachdem nun der Audio-Clip in Frames unterteilt wurde und von jedem Frame die MFCC Feature-Vektoren berechnet worden sind, muss nun die Ähnlichkeit zwischen diesen Vektoren

berechnet werden.

Die Ähnlichkeit zweier MFCC Feature-Vectors (Frame V_i und V_j) wird durch einfache Vektor-Korrelation berechnet:

$$s_{ij} = \frac{V_i \bullet V_j}{\|V_i\| \cdot \|V_j\|} \tag{1}$$

Um die Zeitabhängigkeit der Musik besser zu erfassen, bezieht man auch die benachbarten Frames in die Similarity Measure mitein (siehe Abbildung 4), d.h. man summiert die letzten und nächsten m frames, gewichtet mit $[w_{-m}, ..., w_m]$ (z.B. ein Rechteckfenster):

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

Die Transition Probability von Frame i nach j hängt nun von der Similarity zwischen Frame i+1 und j ab und ist definiert durch:

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

A dient dabei zur Normalisierung, damit $\Sigma P_{ij} = 1$.
σ ist ein Skalierungsparameter: großes σ betont die besten Übergänge, ein kleineres σ bedeutet größere Vielfalt, jedoch eine niedrigere Transition Probability.

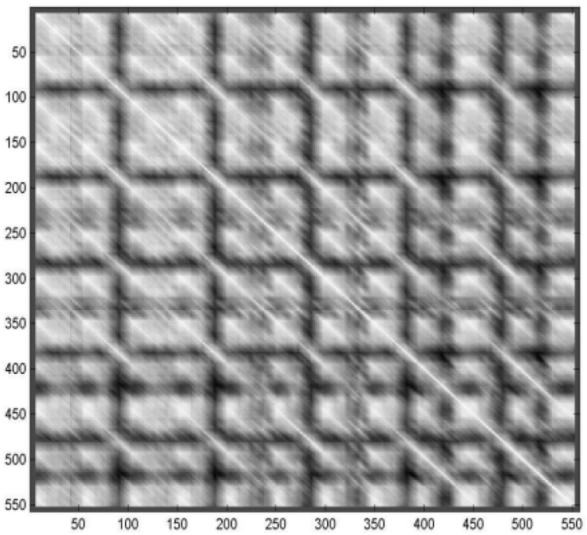

Abbildung 4: Bsp. einer Similarity Matrix welche S'_{ij} für einen Audio-Clip mit 550 Frames repräsentiert. Je heller, desto größer ist die Ähnlichkeit. Illustriert ebenfalls die Selbstähnlichkeit von Musik! (Quelle: [3])

2.3 Sub-Clip Extraktion

Mit Hilfe der Similarity-Matrix (beinhaltet die Transition Probability bzw. Similarity zwischen allen Frames - siehe Abbildung 4) wird nun der Novelty Score berechnet. Dieser gibt an, wie viel sich gegenüber den vorherigen Frames verändert hat.
An den Maxima des Novelty Score, die über einem gewissen Threshold liegen, wird der Audio-Clip nun in Sub-Clips unterteil.

2.3.1 Berechnung des Novelty Score

Die Berechnung des Novelty Score entspricht der Edge-Detection im Image Processing und soll nun an einem Beispiel demonstriert werden.
Z.B. man hat einen Audio-Clip mit 2 total gegensetzlichen Sub-Clips (in Realität nicht möglich - nur zur Illustration), welche jeweils N Frames lang sind.
Dadurch ergibt sich eine Similarity-Matrix der folgenden Form:

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

wobei I_N die NxN Einheitsmatrix darstellt.
Die Hauptdiagonale (=1) entspricht der hohen Selbstähnlichkeit bzw. der Autokorrelation der Sub-Clips und die Nebendiagonale entspricht der geringen Cross-Korrelation.
Diese Matrix wird nun mit der folgenden Kernel-Matrix K korreliert:

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

Da K eine ähnliche Struktur wie S besitzt, korreliert S mit K entlang der Diagonale, d.h. das Ergebnis ist 1 nur an der Position N der Hauptdiagonale, sonst überall 0 (entspricht Edge Detection in Image Processing).
Dies bedeutet, dass ab Frame N etwas neues passiert, in unserem Fall ist hier der zweite, entgegengesetzte Sub-Clip.
Daraus folgt, dass Wert N der Hauptdiagonale als Novelty Scores des Frames N verwendet werden kann, was zur folgenden Definition des Novelty Score führt (für Frame i):

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

Zuletzt wird an den lokalen Maxima der Novelty Kurve, welche über einem gewissen Threshold liegen, der Audio-Clip in Sub-Clips unterteilt (siehe Abbildung 5).

2.3.2 Similarity, Transition-Probability zwischen Sub-Clips

Nun wird wiederum, ähnlich wie in Abschnitt 2.2, Similarity und Transition-Probability zwischen den einzelnen Sub-Clips berechnet.

Die Ähnlichkeit zwischen Sub-Clips wird mit folgender Formel bestimmt:

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

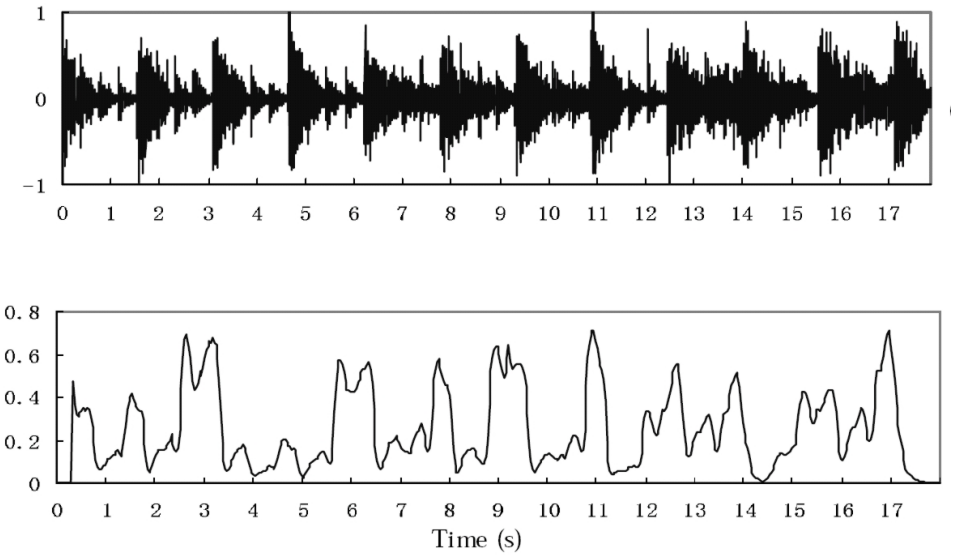

Abbildung 5: Novelty Score Kurve mit zugehöriger Waveform. An den Maxima wird Waveform in Sub-Clips unterteilt. (Quelle: [3])

hier enthält Sub-Clip i M Frames und beginnt bei Frame i, Sub-Clip j enthält N Frames und beginnt bei Frame j und $M < N$, für w_k wurde ein symmetrisches Rechteckfenster verwendet.

Es ist nun wiederum aussagekräftiger, wenn man auch die benachbarten Sub-Clips miteinbezieht (siehe Gleichung 2):

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

Die Transition-Probability von Sub-Clip i zu j wird bestimmt durch $S''_{i+1,j}$ und kann berechnet werden gleich der Transition-Probability zwischen Frames (siehe Gleichung 3).

3 Unconstrained Synthesis

Bei der Unconstrained Synthesis wird, wie bereits erwähnt, ein beliebig langer Audiostream generiert, d.h. es muss eine Sub-Clip Reihenfolge bestimmt werden und danach können noch zusätzliche Effekte hinzugefügt werden.

3.1 Bestimmung der Sub-Clip Reihenfolge

Das Problem bei der Bestimmung der Reihenfolge ist, dass man nicht einfach die maximale Transition-Probability P_{ij} von einem Sub-Clip zum Nächsten nehmen kann, da sonst Loops im Audio-Clip entstehen.
Deshalb ist man in [4], nach Experimenten, auf folgende Formel für das Nachfolgeframe j gekommen:

$$j \in \{j | P_{ij} > P_0\} \bigcap j \notin (l - r, l + r) \tag{9}$$

hierbei ist P_0 ein Threshold, wenn dieser klein gewählt wird bedeutet es, dass viel Variationen entstehen können.
$(l - r, l + r)$ ist eine Schranke, die direkte Vor-, Nachfolge Sub-Clips ausschließt, r wird dabei für jede Auswahl in einem bestimmten Zufallsbereich neu bestimmt.

3.2 Zusätzliche Effekte

Um mehr Variationen zu bekommen, werden folgende Effekte hinzugefügt: Time-Scaling, Pitch-Shifting und Amplitudenveränderung.
Die Bestimmung der Kontrollparameter kann hierbei zufällig oder manuell geschehen.

Pitch-Shifting ist implementiert mit TD-SOLA (Time Domain Synchronous Overlap And Add, siehe [6]). Dabei ist darauf zu achten, dass keine abrupten Sprünge entstehen, deshalb wird zwischen Gruppen mit unterschiedlichem Pitch interpoliert.

Weiters können Sub-Clips überlagert werden, was jedoch machmal zu unerwünschten Nebeneffekten führen kann (z.B. bei Pferdegalopp: wenn Sub-Clips überlagert werden, klingt es wie eine Gruppe von Pferden ...).

4 Constrained Synthesis

Constrained Synthesis, bzw. Audio Texture Restoration, bedeutet, dass ein Fehler in einem Audiosignal detektiert und ausgebessert wird.
Im Gegensatz zu anderen Restaurations-Algorithmen eignet sich Audio Texture Restoration für längere Fehler bzw. Drop-Outs (größer einer Sekunde).
Dabei müssen zuerst die Eckpunkte des Fehlers detektiert werden und anschließend kann eine neue Frame-Abfolge im fehlerhaften Bereich bestimmt werden.

4.1 Detektion des Fehlers

Der fehlende Teil (Drop-Outs werden hier als Fehler angenommen) wird anhand der Novelty Score (siehe Abschnitt 2.3.1) detektiert:
Am Anfangspunkt i_0 und Endpunkt j_0 des Fehlers ist der Novelty Score maximal und im fehlenden Teilstück Null (Drop-Out), dadurch kann der Bereich des Drops-Outs bestimmt werden.

4.2 Bestimmung der Frame-Abfolge

Bei der Constrained Synthesis werden Frames anstatt Sub-Clips als Syntheseeinheit verwendet, da die Länge der Sub-Clips immer verschieden ist und nicht an die Länge des fehlerhaften Bereichs angepasst werden kann (müssten abgeschnitten werden).

In der optimalen Frame-Abfolge soll die Transition-Probability von Frame i_0 zu j_0 optimiert werden:

$$\max P\left(i_0 \rightarrow j_0\right) = P_{i_0,i_0+1} \cdot P_{i_0+1,i_0+2} \cdots P_{j_0-2,j_0-1} \cdot P_{j_0-1,j_0} \tag{10}$$

unter der Voraussetzung: $P_{i,i+1} > p_0, i_0 \leq i \leq j_0 - 1$, p_0 ist ein Threshold

Das Problem dabei ist, dass der Lösungsraum sehr groß ist und es deshalb zu rechenintensiv wäre, die optimale Lösung zu finden.
Deshalb wird mittels Dynamic Programming (ein Optimierungsverfahren um die Laufzeit eines Algorithmus zu minimieren - siehe [7]) eine minimale Cost-Function von i_0 zu j_0 bestimmt.

5 Applikationen und Evaluierung

In [4] wurden einige Klangbeispiele [1] als Beispiele angegeben:

- Klangbeispiele Unconstrained Synthesis:
  Einigermaßen akzeptable Ergebnisse für Wind etc., aber unbrauchbar für Musik.
  Man hört ebenfalls, dass der Einsatz von diversen Effekten (speziell Pitch-Shifting) manchmal zu unerwünschten, unnatürlich klingenden Nebenerscheinungen führen kann.

- Klangbeispiele Constrained Synthesis:
  Für Wind, Wasserrauschen, etc. wurden ganz akzeptable Ergebnisse erreicht.
  Bei Musik ist es wiederum problematisch, da natürlich Rhythmus und Tonphrasen nicht richtig eingehalten werden.

Weiters wurden in [4] psychoakustische Versuche mit Personen durchgeführt. Diese mussten zwei Parameter (*smoothness* und *variety*) bei Unconstrained Synthese ohne jegliche Instruktion bewerten.
Meiner Meinung nach ist das Ergebnis aber leider überhaupt nicht aussagekräftig, da nämlich nicht angegeben wurde, welche Klänge (ob Musik, Wind, ...) von den Versuchspersonen evaluiert worden sind, was natürlich das Versuchsergebnis unbrauchbar macht.

5.1 Limitationen

Schlechte Ergebnisse entstanden für Klänge mit komplexerer Struktur bzw. für Musik und/oder Gesang ist die Methode meist überhaupt nicht geeignet.
Mögliche Gründe sind, dass MFCC als einziges Feature zur Analyse verwendet wurde, welche wenig Rücksicht auf Tonhöheninformationen nimmt, was zu Tonhöhen-Diskontinuitäten führen kann.
Außerdem wurde kein Beat-Tracking durchgeführt und deshalb kann die rhythmische Struktur von beat-orientierter Musik meist nicht beibehalten werden.

Literatur

[1] Francois Pachet Aymeric Zils. Musical mosaicing. Conference on Digital Audio Effects 2001. http://www.csis.ul.ie/dafx01/proceedings/papers/zils.pdf (zuletzt aufgerufen 19.1.2006).

[2] Arno Schödl et al. Video textures. SIGGRAPH 2000. http://www.cc.gatech.edu/cpl/projects/videotexture/SIGGRAPH2000 (zuletzt aufgerufen 19.1.2006).

[3] Liu Wenyin Hong-Jiang Zhang Yi Mao Lie Lu, Stan Li. Audio textures. 2002. http://research.microsoft.com/users/llu/Publications/ICASSP02_AT.pdf (zuletzt aufgerufen 19.1.2006).

[4] Liu Wenyin Liu Lu. Audio textures: Theory and applications. 2004. IEEE Transaction On Speech And Audio Processing.

[1] downloadbar unter http://research.microsoft.com/~llu/AudioTextures, zuletzt aufgerufen am 6.2.2006

[5] Beth Logan. Mel frequency cepstral coefficients for music modeling. Cambridge Research Laboratory, Compaq Computer Corporation.

[6] Piotr Majdak. Zeit-frequenz-verarbeitung. http://iem.at/~majdak/alg/index.html (zuletzt aufgerufen 19.1.2006).

[7] Wikipedia. Dynamic programming. 2006. http://en.wikipedia.org/wiki/Dynamic_programming (zuletzt aufgerufen 4.2.2006).