無音動画に対する効果音貼付けシステムの開発

鈴木 喜也[†1,a)] 岡部 誠[†1,†2,b)] 尾内 理紀夫[†1,c)]

概要：本研究では, 無音の動画に対して効果音を貼り付ける作業を効率化するシステムの開発を行った. 本システムはガラスの割れる音や水のはねる音といった持続性のある効果音に注目し, これらの効果音を, 既存手法よりも劣化を抑えて伸長するアルゴリズムを考案した. また, システムとしてドラッグ＆ドロップによる操作だけで効果音の伸長や発音位置調整を行うユーザインタフェースも実装した. 本研究により効果音編集の経験のないユーザでも容易かつ効率的に効果音の貼り付け作業を行うことが可能となる.

キーワード：動画製作, 音処理, 効果音編集, 効果音の伸長

The System Development of Sound Effect Synthesis for Soundless Video

Suzuki Nobuya[†1,a)] Okabe Makoto[†1,†2,b)] Onai Rikio[†1,c)]

Abstract: In this paper, we developped a system to synthesize the sound effect on soundless video. In this system, we targeted the extension procesing of sound effect that lasts a few seconds, such as glass breaking and water splitting. And our extension algorithm can extend the sound effects that is difficult to stretch in the existing method with less degradation. In the user interface , user can perform sound positioning and extension of sound effects just repeat the drag-and-drop. By our system , users without sound effect editing can synthesis the sound effect more easily.

Keywords: Video Production, Sound Processing, Sound Effect Editing, Sound Effect Extension

1. はじめに

効果音の編集作業は動画製作において必ず通らなければならない工程であり, 同時に動画の質を左右する重要な工程でもある. 近年, ニコニコ動画や YouTube に代表される動画コンテンツの普及によりアマチュアの動画製作者が日々増え続けている. これに伴い動画製作を支援するツールに対する需要が増加している.

ここでアマチュアの動画製作工程に目を向ける. アマチュアの動画製作者はまず画像編集用のソフトウェアを利用して画像をアニメーションさせ, 無音状態で動画を出力する. 本研究ではこの状態の動画を無音動画と呼ぶ. 次に別のソフトウェアを用いて音の編集を行い, 無音動画に音を貼り付けて動画を完成させる. 多くの動画製作の現場では, この 2 つの工程で動画を製作している. 無音動画に貼り付ける音として, 音声, 音楽, 効果音が挙げられる. このうち音声と音楽の編集に関する研究は盛んに行われているが効果音の編集に関する研究は少なく, 特に無音動画に容易かつ効率的に効果音を貼り付ける手法は未開の領域である. そこで本研究では, 効果音の貼り付けを対象として扱うこととした.

効果音の貼り付け作業における労力は, 効果音の発音位置を調整する作業と効果音を無音動画に適した長さに伸長する作業の 2 点が最も大きい. 特に効果音を伸長する作業は, 適用するフィルタの選択やパラメータの設定を誤ると望む長さに伸長できない, 音が劣化してしまうといった難点があり, 動画製作者の経験に依存する作業であると言える. このためアマチュアの動画製作者は効果音にフィルタを適用して試聴し, 望む効果音が得られなければやり直すという試行錯誤を繰り返すことで効果音の伸長作業を行っ

†1 現在，電気通信大学
Presently with The University of Electro-Communications
†2 現在，JST CREST
a) suzuki@onailab.com
b) m.o@acm.org
c) rikioonai@gmail.com

ている．このため, 効果音の伸長作業は敷居が高いものとなっている．中でも水のはねる音やガラスの破壊音といった持続性のある効果音は, 波形の性質上既存の手法により編集すると音の劣化が激しく, 扱うことが困難である．

そこで本研究ではこの現状を解決するため, 持続性のある効果音の劣化を抑えて伸長するアルゴリズムを考案した．またこのアルゴリズムを用いて, 効果音の編集に不慣れなユーザが持続性のある効果音を適切な長さに伸長し, 無音動画に貼り付けられるシステムを開発した．

以降は第 2 章において効果音の分類を行い, 本研究で扱う効果音について述べ, 第 3 章で音の伸長における既存手法を挙げる．第 4 章では本研究で提案する効果音の伸長アルゴリズムの概要について述べ, 第 5 章でアルゴリズムの詳細を述べる．第 6 章では本システムのインタフェースとユーザの行う操作を示し, 第 7 章でシステムの評価実験について述べ, 第 8 章にてまとめる．

2. 効果音の分類

初めに動画製作で使用される効果音の分類を行い, 本研究で扱う効果音の対象を設定する．

本研究では動画製作で使用される効果音を繰り返し音, 瞬間的な音, 持続性のある音の 3 種類に分類した．表 1(次ページ) に効果音の分類表を, 図 1 に分類した 3 種類それぞれのスペクトログラムの例を示す．スペクトログラムは 22kHz 以下の周波数成分を表示している．最も強い成分は白色で表され, 赤色から青色になるにつれて弱い成分であることを表している．図 1 を元にそれぞれの音の特徴について述べる．

- 繰り返し音 (図 1a)

  風の音や車の走行音, 鍋のグツグツ音などがこの種類に該当する．図 1(a) では風の音を例として挙げた．スペクトログラムを見ると, 低周波に強い成分 (白色で表された部分) が同じ形状で繰り返し現れていることが分かる．このように繰り返し音は周期性をもち, 音の連続性が大きい点が特徴である．

  繰り返し音については, 飯島らの研究 [1] により周期性を分析し, 逆再生を用いることで伸長が可能であることが示されている．このため本研究で繰り返し音は扱わないものとした．

- 瞬間的な音 (図 1b)

  足音や物体の衝突音がこの種類に該当する．図 1(b) ではスニーカーの足音を例として挙げた．スペクトログラムの形に現れているように, この種類の効果音は持続時間が一瞬しかない点が特徴である．そのため動画製作においてはミリ秒単位で発音位置を調整し, 動画内の物体の動きと効果音の発音タイミングを一致させる必要がある．

  我々は先行研究 [2] において, 瞬間的な音を半自動的に無音動画に貼り付けるシステムの開発を行い, 一定の成果を得た．そのため本研究で瞬間的な音は扱わないものとした．

- 持続性のある音 (図 1c)

  ガラスの破壊音や水のはねる音, 爆発音といった繰り返しがなく, 数ミリ秒から 5 秒程度持続する音がこの種類に該当する．(図 1c) ではガラスの破壊音を例として挙げた．

  この種類の効果音はスペクトログラムに見られるように, 広い周波数領域にわたって不規則に音の成分が現れる点が特徴である．

  無音動画にこの種類の効果音を貼り付ける場合, 無音動画内の物体の動きに合わせて効果音を伸長させる必要がある．しかし前述した 2 種類の効果音とは異なり, この種類の音を簡単に伸長する手法は存在しない．そこで本研究では持続性のある効果音に的を絞り, この種類の効果音を簡単に伸長するアルゴリズムを考案した．

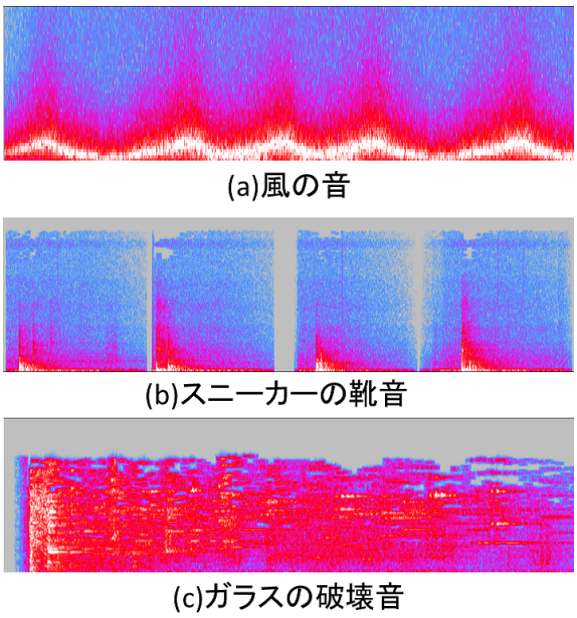

図 1 動画製作で用いられる効果音の例

3. 関連研究

現在最も広く利用されている音の伸長アルゴリズムとして,SOLA(Synchronous OverLap-Add)[3] が挙げられる．SOLA は Rabiner らにより提唱された手法であり, 重畳加算法 (OLA,OverLap-Add method) と呼ばれる演算手法を用いて波形を伸長するタイムストレッチ系のアルゴリズムである．

SOLA による波形の伸長は以下の手順で行われる．

(1) 入力された波形の分割

まず入力された波形を一定の長さで細かく区切る．

表 1 効果音の分類

	代表例	特徴	対応する手法
繰り返し音	風の音, 車の走行音, 鍋のグツグツ音	音に周期性があり連続性が大きい	飯島らによる手法 [1]
瞬間的な音	足音, 衝突音	発音時間が数ミリ秒と非常に短い	先行研究における手法 [2]
持続性のある音	ガラスの破壊音, 水のはねる音, 爆発音	周波数の広い領域に不規則な音が現れる	提案手法

区切る長さは入力された波形の基本周期である.

波形の基本周期の算出には, まず入力された波形全体をフーリエ変換してパワースペクトル (波形の振幅の 2 乗を時間軸に沿って並べたもの) を得る. 次にこのパワースペクトルを逆フーリエ変換して得られる, 自己相関関数の極大値を求める. 波形の先頭からこの極大値までの時間を基本周期として扱う.

(2) (1) で区切られた各区間に対してフーリエ変換を適用する.

(3) 分割された各区間の伸長

フェードインとフェードアウトを繋げた形状の窓関数を用意する. フェードインは音の開始位置から徐々に音量が大きくなるフィルタ, フェードアウトは音の終了位置に向けて徐々に音量が小さくなるフィルタである. この窓関数にもフーリエ変換を適用し,(2) でフーリエ変換された各区間それぞれに掛け合わせる. その後, 窓関数が掛け合わされた各区間に対して逆フーリエ変換を適用する. 窓関数の長さを N, 各区間の長さを L とすると, フーリエ変換された各区間と窓関数を掛け合わせたデータを逆フーリエ変換して得られた各区間の長さはそれぞれ $L+N-1$ に伸長される.

(4) 伸長された部分を重ねながら連結

(3) で得られた, 伸長された各区間の波形を重ねあわせながら連結する. この時重ねあわせる長さを調整することにより, 任意の長さに音を伸長することが可能となる.

SOLA を改良したアルゴリズムの代表例として PSOLA と WSOLA が挙げられる.PSOLA(Pitch SOLA)[4] は波形の周期の 2 倍の大きさの窓関数を用いることにより, ピッチを維持したまま波形を伸長するアルゴリズムである. また,WSOLA(Waveform Similality OLA)[5] は区切られた各区間に関して, 前後の区間との連続性を保つように窓関数を構成することで区切り毎の繋ぎ目が滑らかな伸長処理を行うものである.

SOLA をベースとしたアルゴリズムは近年においても改良が続けられており,Shahaf Grofit らによる研究 [6] では窓関数の長さを 2 倍にし, 重畳加算法を適用しない区間を設定することで伸長された音の音質の向上が行われている.

これらの手法は現代で最も主流なものであり Adobe Audition[*1] をはじめとする多くのソフトウェア上で用いられている. しかし, これらの手法はスピーチや音楽を伸長することを目的として考案されたものであり, 音の性質が大きく異なる本研究で扱う持続性のある効果音にそのまま適用することは難しい. 既存のタイムストレッチ系アルゴリズムにより伸長を行うことができる音には以下の特徴が求められる.

- 音が長い

  スピーチや音楽の長さは 1 分を超えるものが多い. そのため入力された音を伸長する場合, 要求される長さは入力された音の長さの 2 倍程度までにとどまる. Shahaf Grofit らの研究においても,2 倍に伸長した音楽やスピーチの音質が評価対象とされている. しかし本研究で扱う効果音には長さが 1 秒未満の短い音も存在するため, 入力される効果音によっては 2 倍を超える伸長を行う必要も生じる.

- 周期音を含む

  周期音とは音の基本周波数と, 基本周波数を整数倍した音で構成される音である.

  図 2 にスピーチ [*2] と音楽 [*3] のスペクトログラムを示す. スピーチ (図 2a) と音楽 (図 2b) のスペクトログラムから, どちらも低周波に強い成分の音が顕著に存在している事がわかる. この強い成分には人の声や楽器の基本周波数が含まれており, 強い成分が上下することは音程の変化を表している.

  また高周波数の領域に存在する成分は, 殆どが人の声や楽器の基本周波数の倍音で構成されている. このように基本周波数に基づいた周波数成分の構成をもつため, スピーチや音楽からは波形の基本周波数の推定が可能である. タイムストレッチ系アルゴリズムにおいて, 波形を区切る長さに用いる基本周期は基本周波数の逆数をとったものである. このため, スピーチや音楽はタイムストレッチ系アルゴリズムで劣化の少ない伸長を行うことができる.

図 3 にガラスの破壊音と水をこぼす音をタイムストレッチ系アルゴリズムにより伸長したスペクトログラムを示す. それぞれ上段は原音, 中段は原音を 2 倍に伸長した音, 下段は原音を 4 倍に伸長した音のスペクトログラムである.

図 3A, B は伸長されたことによる変化が特に顕著となった区間である. 図 3A', A" と B', B" は A, B の区間のスペクトログラムが伸長されたものであるが, 伸長後は同じ形状のスペクトログラムが細かく何度も連続している. この

[*1] http://www.adobe.com/jp/products/audition.html

[*2] オバマ大統領の勝利演説 (2008 年 11 月) の一部を引用:http://www.youtube.com/watch?v=BhpsMCkRUjI

[*3] ザ・ビートルズのハロー・グッドバイの一部を引用

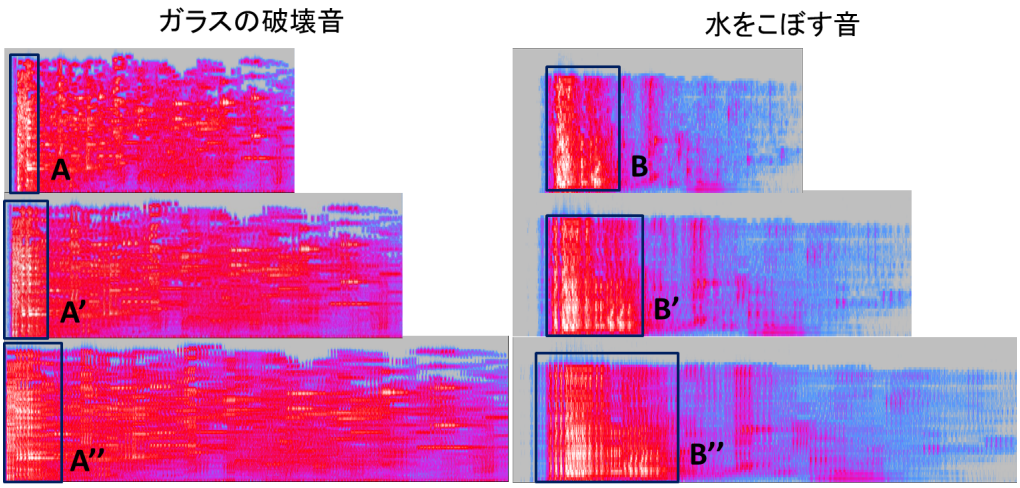

図 3 タイムストレッチ系アルゴリズムによる伸長

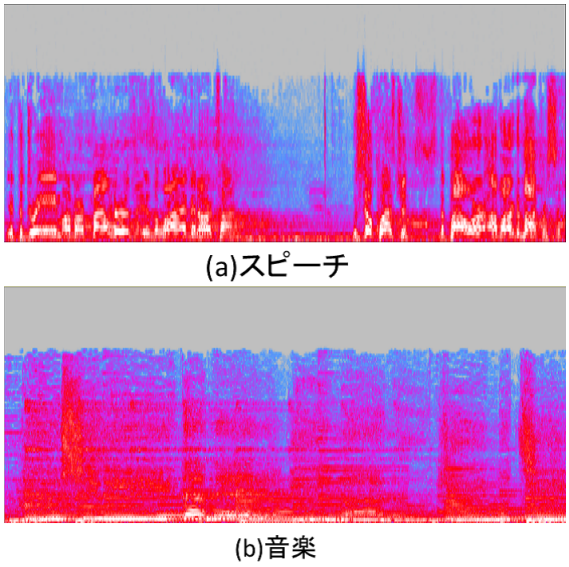

図 2 スピーチと音楽のスペクトログラム

ようにタイムストレッチ系アルゴリズムにより持続性のある効果音を 2 倍以上に伸長する場合, 効果音全体で同じ音が細かく連続し, 原音が大きく劣化する.

これに対し本研究では, 原音の劣化を抑えて持続性のある効果音の伸長を行うアルゴリズムを考案した. アルゴリズムの概要は第 4 章, アルゴリズムの詳細については第 5 章にて述べる.

4. 伸長アルゴリズムの概要

本章では, 持続性のある効果音をユーザの指定した長さに伸長するアルゴリズムの概要を述べる.

図 4 に 2 秒の効果音を本アルゴリズムにより 6 秒まで伸長した際の波形の変化を示す. 本アルゴリズムは効果音の原音を受け取り, ユーザが伸長後の長さを入力として与えると以下の手順で効果音の伸長を行う.

(1) 適用する処理の分岐

ユーザから与えられた伸長後の長さにより処理を分岐させる. 入力された伸長後の長さを L, 効果音の原音の長さを l としたとき, 伸長率を L/l と定める.

この伸長率が 2.0 未満, つまり 2 倍未満の伸長処理を行う場合は既存のタイムストレッチ系アルゴリズムによる伸長を行い終了する. そうでない場合は (2) 以降の処理を適用する.

(2) ピーク部分を除去した波形の生成

効果音を 2 倍以上に伸長する場合, 本アルゴリズムは効果音の音量が最も大きくなる部分の前後 (ピーク部分と呼ぶこととする) を原音から除去した波形を生成する. 図 4(a) では赤枠で囲われた, 音量が最も大きい部分を除去した波形を生成している. ピーク部分の幅の定義と除去手法は第 5.1 節で述べる.

(3) 波形の重ね合わせ

効果音の原音に (2) で生成したピーク部分を除去した波形を重ね合わせることで効果音を伸長する. 波形を重ね合わせた時, 伸長後の効果音の長さを l', ユーザから与えられた伸長後の長さを L とし, 新しい伸長率 L/l' を算出する. この値が 2.0 未満であった場合,(4) に進む. そうでない場合は (3) の最初に戻り, 波形の重ね合わせによる伸長を繰り返す.

図 4(b) では図 4(a) で得た波形を原音の後半部分に重ねることで原音を伸長させている. 伸長処理の詳細は第 5.2 節で述べる.

(4) タイムストレッチによる伸長

(3) で伸長された波形に対しタイムストレッチ系アルゴリズムを適用し, ユーザの入力した長さまで伸長を行い終了する. 図 4(c) では図 4(b) において伸長された波形を 1.25 倍に伸長することにより, 長さ 6 秒の波形を得ている.

図 4　伸長アルゴリズム概要

5. アルゴリズムの詳細

本章では第 4 章で述べた伸長アルゴリズム内で行う処理の詳細を述べる. 以降アルゴリズムの説明を簡単化するため, 効果音を標本化して得られる標本値 1 つを音の最小単位とみなし, これが連続して X 個つながったものを標本値 X 個のデータと呼ぶこととする. 標本化の際のサンプリング周波数は動画で最も多く使われる 44.1kHz で統一した.

5.1 ピーク部分を除去した波形の生成

効果音の原音からピーク部分を除去した波形を生成する手法について述べる. この処理では入力として効果音の原音を受け取り, 以下の手順によりピークを除去した波形の生成を行う.

(1) 閾値の設定

ピーク部分の開始位置と終了位置を判断するための閾値 α の設定を行う. まず入力された原音の全標本値の絶対値をとり, これらの絶対値の平均値 μ, 標準偏差 σ を算出する. 次にこれらの値を用いて閾値 α を $\alpha = (\mu + 2\sigma)$ と設定する. 一般に, 平均と標準偏差の 2 倍の和を超える値を持つ要素の数は全体の要素の 25% を超えない.

(2) ピーク部分の終了位置の探索

入力された原音のピーク部分の終了位置を探索する. まず入力された効果音の先頭を波形を切り抜く基準点に設定し, この基準点から標本値 1000 個のデータを抽出する. 抽出したデータに含まれる各標本値の絶対値をとり, それらの平均値を β とおく. $\alpha < \beta$ であれば,β を算出した区間の音量が原音全体の音量の上位 25% に入ったと判断し, この時の基準点をピーク部分の開始位置とみなす. そうでない場合は基準点を標本値 250 個分ずらし, 同様に α との比較を行う.

ピーク部分の開始位置に入った場合, この点からさらに基準点を標本値 250 個分ずらし, 同様に β を算出する. $\beta < \alpha$ であれば, この時の基準点をピーク部分の終了位置とみなし T とおく. そうでない場合標本値 250 個分基準点をずらし, 同様に β を算出して α との比較を行う.

一度ピークの開始位置の算出を行うのは, 効果音がピーク部分に向かって徐々に音量が大きくなる形状をしていた場合, ピーク部分の終了位置の判定だけではうまくピーク部分を除去することができないためである.

(3) ピーク以降の波形を抽出

(2) で得られた T 以降の波形を切り抜き, ピーク部分を除去した波形として扱う.

図 5 にガラスの破壊音と爆発音を例にピークの除去処理を施す前と後の波形を示す. 各波形の形状から, 本節の処理により原音のピーク部分が除去されていることがわかる.

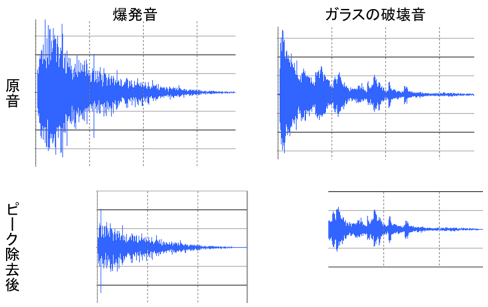

図 5　ピーク部分の除去による波形の変化の例

5.2 重ね合わせ処理

効果音の原音と第 5.1 節で生成したピーク部分を除去した波形を重ねあわせる処理について述べる. 一般的に波形を重ねあわせるフィルタとして, ディレイフィルタが知られている. ディレイフィルタは効果音 1 つに加えて, 重ね

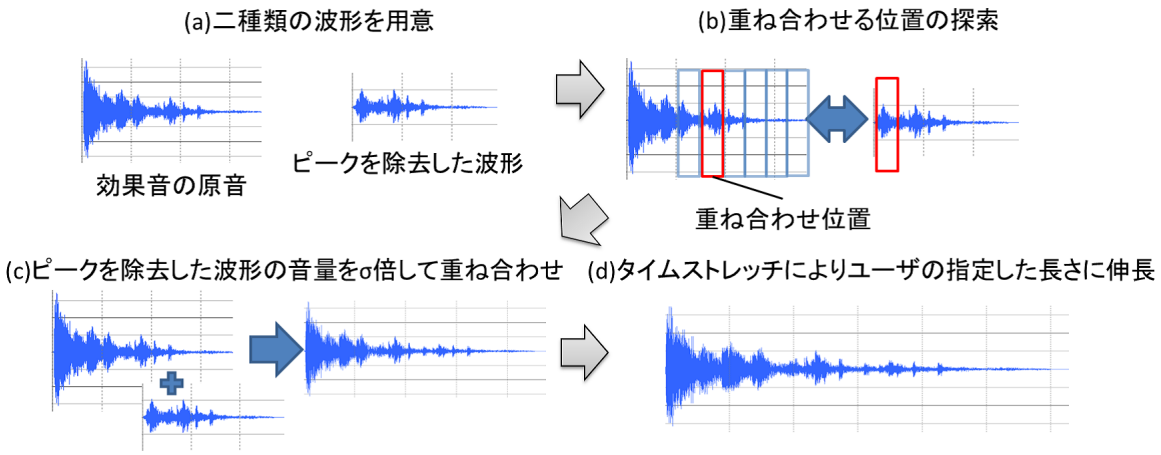

図 7　重ねあわせによる波形の変化の例

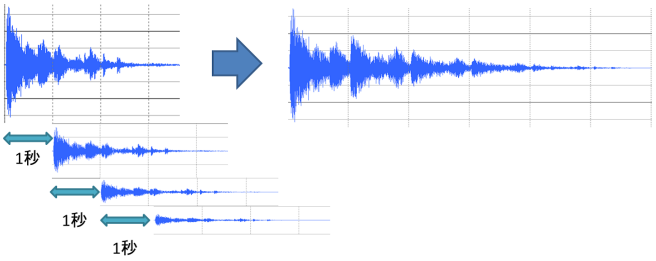

図 6　ディレイフィルタの例

あわせ毎の減衰率 λ, 重ねあわせる間隔 δ(秒), 重ね合わせ回数 κ(回) の 3 つのパラメータを必要とする.

図 6 にガラスの破壊音に対して $\lambda = 0.5$, $\delta = 1.0$, $\kappa = 3$ のディレイフィルタを適用した例を示す. ディレイフィルタは与えられた効果音に対し, 以下の処理を適用する.

(1) 波形を重ねあわせた回数を i とし, 初期値に 0 を設定する. また, 入力された効果音の全標本値を $\lambda^{(i+1)}$ 倍した波形を生成する.

(2) (1) で得られた波形を効果音の開始位置から $(i+1) \times \delta$ 秒後の位置に重ね合わせる.

(3) $i < \kappa$ であれば i を 1 増加させ,(1) に戻る. そうでなければ終了する.

ディレイフィルタは波形の重ねあわせを行うことができるが, ディレイフィルタを本アルゴリズムの重ね合わせ処理に用いる場合, 以下の問題点がある.

- 出力される効果音の長さを指定することができない. さらに, 各パラメータをどのように設定すれば目的の長さに効果音を伸長できるかがユーザに分かりづらい.
- 重ねあわせに原音以外の波形を用いることができない. そのため本研究で扱う効果音のような音量の大きい音に適用すると突然音が大きくなり違和感が生じる.

そこで本アルゴリズムでは, 次に述べるような重ね合わせ処理を新たに考案した. この重ね合わせ処理でユーザが入力するパラメータは伸長後の長さのみであるため, ユーザが直感的に効果音の伸長を行うことができる. さらに重ねあわせる音に第 5.1 節において生成したピーク部分を除去した波形を用いることで, 音量の極端な変化を抑えて効果音を伸長する.

図 7 に波形の重ね合わせ処理における波形の変化を示す. 図 7(a) は重ね合わせ処理に用いる 2 種類の波形を示したものである. この処理には, 効果音の原音と原音のピーク部分を除去した波形を用いる. ユーザから伸長後の長さが入力として与えられると以下の手順により効果音の伸長を行う.

(1) 減衰率の設定

まずピーク部分を除去した波形の音量を調整する値 λ を設定する ($\lambda < 1.0$). 本アルゴリズムでは複数の効果音に対して実験を行った結果, 伸長率が 3 倍未満の場合は $\lambda = 0.6$, 3 倍以上 5 倍未満の場合は $\lambda = 0.7$, 5 倍以上の場合は $\lambda = 0.8$ とした. また, 波形を重ねあわせた回数を i とし, 初期値に 0 を設定する.

(2) 重ねあわせ位置の探索

原音とピーク部分を除去した波形を重ねあわせる位置の探索を行う. ピーク部分を除去した波形の先頭から標本値 1000 個のデータを抽出し, このデータに含まれる全標本値の絶対値をとる. この絶対値の平均値を a とおく. 次にデータを切り抜く基準点を原音の末尾に設定し, この基準点から標本値 1000 個のデータを抽出する. このデータに含まれる全標本値の絶対値をとり, この絶対値の平均値を b とおく.

$a \times \lambda^{(i+1)} \geq b$ であれば, この時の基準点を τ とおいて (3) に移る. そうでない場合, 基準点を標本値 250 個分原音の先頭に向かってずらし, 再度 b を算出して a と比較を行う.

図 7(b) はピーク部分を除去した波形の赤枠で囲われた部分を原音と比較し, 重ねあわせる部分を探索する様子を表している. ピーク部分を除去した波形の赤枠で囲われた部分と, 原音の青枠で囲われた部分の音量を末尾から順に比較し, 最終的に原音の赤枠で囲われた部分を重ねあわせ位置として選択する.

(3) 重ね合わせ

ピークを除去した波形の全標本値に $\lambda^{(i+1)}$ をかけて音量を小さくし, (2) で取得した τ の位置にこの波形を重ねあわせる.

図 7(c) では図 7(b) で示した, 原音の赤枠の位置にピークを除去した波形を重ねた様子を表している. 原音のピーク部分を除去した波形を重ねあわせに用い, 重ね合わせる位置を原音の音量から判断することにより, 音量の極端な変化を抑えて波形を伸長することができる.

(4) ユーザの指定した長さとの比較

(3) で得た波形の長さを l, ユーザが入力した伸長後の長さを L とする. $L/l < 2.0$ であった場合, タイムストレッチ系アルゴリズムにより長さ L まで (3) で得た波形を伸長し, 終了する. そうでない場合は効果音の原音を (3) で得た波形に置き換え,i を 1 増加させて (2) に戻る.

図 7(d) では図 7(c) で得られた波形を 1.25 倍に伸長させている. 図 7 の例ではこの一連の処理により,2 秒の効果音を 6 秒まで伸長させた. このように本アルゴリズムでは伸長率が 2 倍を超える伸長にも対応が可能である.

6. ユーザインタフェース

本章では, 本研究で開発したシステムのユーザインタフェースとそれを用いてユーザが行う操作について述べる. 本システムのユーザはまず無音動画を作成し, それに貼り付ける効果音 (複数可) を用意し, 本システムを使用する.

第 6.1 節ではインタフェースの外観とシステムが提供する各インタフェースの役割について述べ, 第 6.2 節ではシステムを用いた効果音の貼り付け作業においてユーザが行う操作を述べる.

6.1 インタフェースの外観

図 8 に本システムのユーザインタフェースを示す. 本システムのユーザインタフェースはプレビューウィンドウ (図 8a), タイムライン (図 8b), 素材ウィンドウ (図 8c) の三種類のウィンドウで構成される.

プレビューウィンドウは無音動画とタイムライン上に配置した効果音の再生機能を有するウィンドウである. ウィンドウ内には動画の再生画面と再生の制御ボタンをもつ.

タイムラインは効果音の発音位置の調整と伸長の操作を受け付けるウィンドウである. タイムライン内は効果音をドラッグ＆ドロップする長方形の領域が 3 つと無音動画の再生位置を操作するトラックバーをもつ. トラックバーのつまみの位置はプレビューウィンドウに読み込まれた無音動画の再生位置と同期しており, つまみを時間軸に沿って動かすと, プレビューウィンドウ内の無音動画の再生位置を変更することができる.

素材ウィンドウは事前にユーザが用意した効果音と無音動画をアイコンで表示したものである. 使いやすさの向上のため, 効果音を表示するウィンドウと無音動画を表示するウィンドウの二種類のウィンドウで表示している. また, ウィンドウ内に表示された音符とフィルムのアイコン 1 つ 1 つがユーザが用意した各効果音と動画に対応している. ユーザはこれらのアイコンから無音動画とそれに貼り付ける効果音の選択を行う.

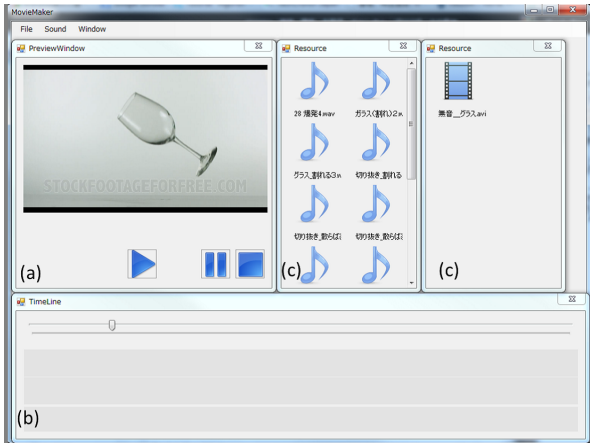

図 8 インタフェースの外観

6.2 ユーザによる操作

図 9 に, 本システムを用いたユーザが行う効果音の貼付け作業の操作を示す. ユーザは以下の操作により無音動画に効果音を貼り付ける.

(1) 効果音をタイムライン上にドラッグ＆ドロップ

ユーザは素材ウィンドウから無音動画に貼り付ける効果音を選択し, タイムライン上の任意の位置にドラッグ＆ドロップする. この時, 効果音をドロップした位置に赤いバー (以下効果音バーと呼ぶ) が生成される.

図 9(a) は, ガラスの割れる音をタイムライン上にドラッグした様子である. 図 9(b,c,d) のタイムライン上に表示されている効果音バーにはユーザがドロップした効果音が格納されている. 効果音バーの位置は効果音の発音位置, 長さは効果音の長さに対応する.

(2) 発音位置の調整

ユーザは効果音バーの左端をつかみ, タイムラインに沿って移動させることで効果音の発音位置を調整する. 効果音バーをタイムラインに沿って移動させている間, プレビューウィンドウ内の無音動画の再生位置が効果音バーの位置に変更される. これにより, ユーザはプレビューウィンドウ内の無音動画を見て効果音を適切な位置に配置することができる.

図 9(b) では, 効果音バーを動かしながらプレビューウィンドウ内の無音動画を確認し, グラスが床に接触

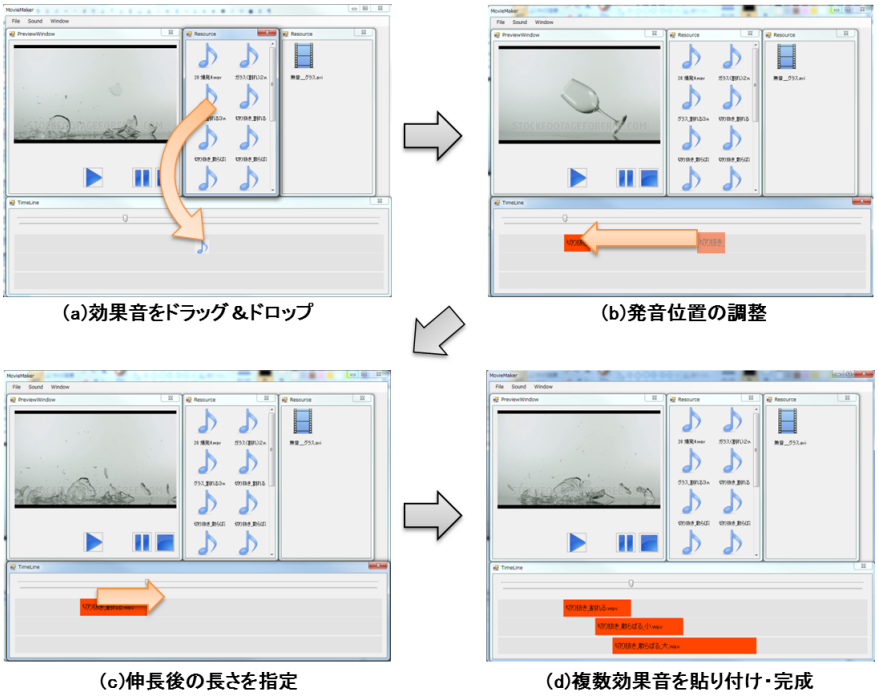

図 9 ユーザによる操作

する瞬間の位置に効果音バーを配置している.

(3) 伸長後の長さの指定

ユーザは効果音バーの右端をつかみ, タイムラインに沿ってドラッグすることで伸長後の効果音の長さを指定する.

図 9(c) は効果音バーの右端をドラッグして効果音バーを引き伸ばす様子を表している. プレビューウィンドウ上の無音動画を確認しながら効果音バーを引き伸ばし, ガラスが散らばるシーンになったことを確認してドラッグを止める. ドラッグを止めた時, システムは効果音バーの長さから効果音の伸長後の長さを算出する. この長さを元に効果音の原音に対して伸長アルゴリズムを適用し, 効果音バーに格納する.

(4) 動画の完成

(1)〜(3) の操作を繰り返すことで無音動画に対して効果音を貼り付ける. 最終的に図 9(d) のように複数の効果音がタイムライン上に載せられ, 長さが調整されることで動画の完成となる.

このようにユーザの行う処理はインタフェース上でのドラッグ＆ドロップのみである. このため効果音の編集に不慣れなユーザであっても, 容易かつ効率的に無音動画に効果音を貼り付けることができる.

7. 評価実験

本研究で開発したシステムを用いて, 無音動画に持続性のある効果音を貼り付ける実験を著者が行った. 実験には効果音を貼り付ける無音動画として, ガラスの割れるシーンの無音動画を 3 種類, コップの水をばらまくシーンの無音動画を 3 種類, 物体の爆発シーンの無音動画を 2 種類の合計 8 種類を用意した. また持続性のある効果音としてガラスの破壊音を 4 種類, 水のはねる音を 3 種類, 爆発音を 5 種類の合計 12 種類を用意した. 比較対象として, 同じ無音動画と効果音を用いて既存のソフトウェア [*4] により効果音を伸長し, 貼り付けた.

図 10 にガラスの散らばる音と水のはねる音を既存のソフトウェアと本システムで伸長したスペクトログラムの例を示す. それぞれ上段が効果音の原音のスペクトログラム, 中段が既存のソフトウェアで伸長した効果音のスペクトログラム, 下段が本システムで伸長した効果音のスペクトログラムである. 図 10A, B は伸長されたことによる, 既存手法と提案手法のスペクトログラムの変化の違いが特に顕著となった区間であり, 図 10A', A" と B', B" は A, B の区間のスペクトログラムが伸長されたものである.

図 10a は 0.5 秒のガラスの散らばる音を 6 倍の 3 秒に伸長した例である. 既存のソフトウェアで伸長したスペクトログラムは原音のスペクトログラムの強い成分 (図 10A) が非常に長く引き伸ばされている (図 10A'). このように音の各成分が無理に長く引き伸ばされた結果, 既存手法で伸長した効果音は原音とはかけ離れた音になった. これに対し本システムで伸長した効果音は, 強い成分が引き伸ばされることなく伸長されている (図 10A"). また, 図 10b は 1.1 秒の水のはねる音を 3 倍の 3.5 秒に伸長した例である. 既存のソフトウェアで伸長したスペクトログラムは, 原音で

*4 比較対象には Adobe Audition, Adobe After Effects, Audacity(http://audacity.sourceforge.net/) を用いた

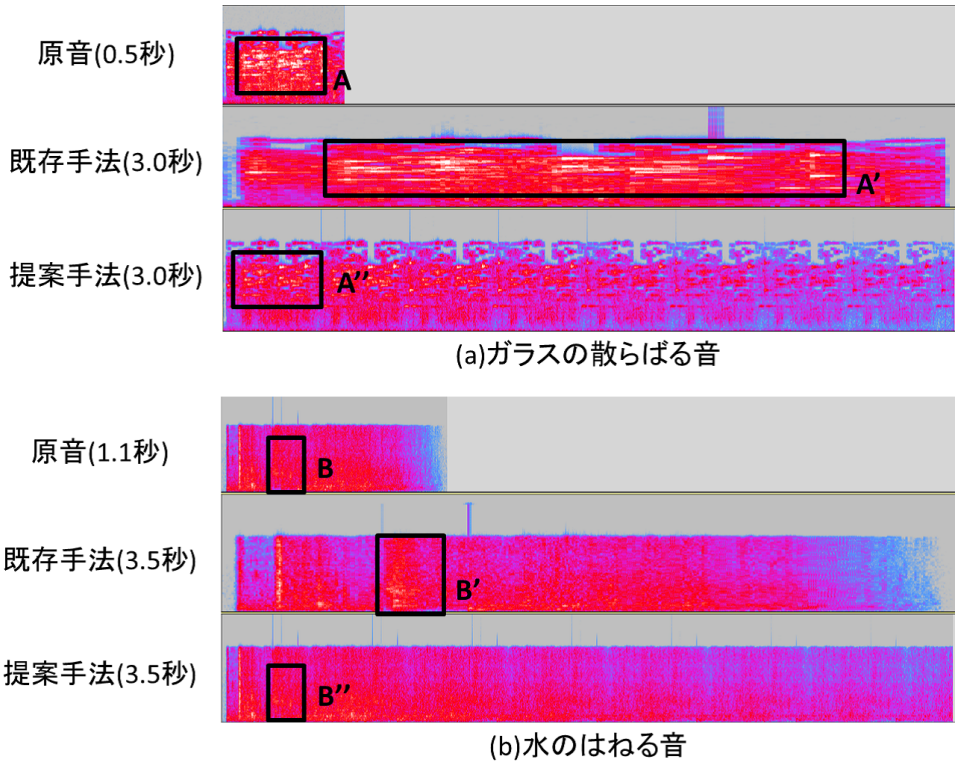

図 10 既存手法とのスペクトログラムの比較の例

微小だった強い成分 (図 10B) が, 何度も細かく連続する形になった (図 10B’). これにより, 既存のソフトウェアで伸長された水のはねる音は細かい音が何度も繰り返す違和感のある結果になった. これに対し本システムで伸長した効果音のスペクトログラムは, 原音の成分がそのまま維持されており (図 10B”), 違和感の少ない結果を得ることができた. また, ガラスの散らばる音の例では原音の長さの 6 倍, 水のはねる音の例では原音の長さの 3 倍と, 既存のタイムストレッチ系アルゴリズムでは音の劣化により実現できなかった長さに持続性のある効果音を伸長することができた.

これら以外の効果音の貼り付けにおいても, 既存のソフトウェアで伸長した効果音の多くは図 10A’ のように効果音の成分が非常に長く引き伸ばされる, 図 10B’ のように音が細かく連続するといった劣化が生じた. これに対し本システムで伸長した効果音は, 劣化を抑えて長い時間に伸長されていることを確認した. また効果音の貼り付けに要した作業時間に関しても, 既存のソフトウェアで効果音の貼り付けを行った場合と比べて本システムを用いて効果音の貼り付けを行った場合は, ドラッグ＆ドロップにより直感的に効果音の伸長と貼り付けを行うことができるため作業時間が大幅に短縮されることを確認した.

しかし用意した 5 種類の爆発音のうち, 本アルゴリズムで伸長することは可能だが, 音が劣化する効果音が 1 種類, 伸長することができない効果音が 1 種類存在した. 図 11 に本システムで伸長した結果, 伸長されたが音が劣化した爆発音 (図 11 上段) と伸長に失敗した爆発音 (図 11 下段) の伸長を行う前のスペクトログラム (図 11 左), 伸長後のスペクトログラム (図 11 中央), 原音のピーク部分を除去したスペクトログラム (図 11 右) を示す. 図 11A, B はそれぞれの爆発音のピーク部分である.

図 11 上段の伸長後のスペクトログラムは, 音量が突然大きくなるスペクトログラムの形が何度も現れており, エコー感が強い音となった. また, 図 11 下段のスペクトログラムは伸長アルゴリズムを適用したにもかかわらずほとんど変化していない.

これらの原因は原音の音量変化の仕方にあると考えられる. 図 11 上段の原音のスペクトログラムは発音時に瞬間的に音量が大きくなり, そこから急速に音が小さくなる形状となっている. この形状の特徴により, 原音のピーク部分が図 11A のように効果音の発音開始付近の非常に短い区間と判定されてしまった. このためピーク部分を除去した波形の発音開始位置の音量が大きく, 重ね合わせ処理でピーク部分を除去した波形が発音開始位置の付近に重ね合わせられてしまい, 重ね合わせる回数が多くなったと考えられる. また図 11 下段のスペクトログラムは, 図 11B で示したピーク部分の後再び音量が大きくなる, ピーク部分が 2 回存在する形状となっている. このためピーク部分を除去した波形には 2 回目のピーク部分がそのまま残っており, 重ね合わせ処理でピーク部分を除去した波形が 2 回目のピーク部分に重ねられることで伸長に失敗したと考えられる.

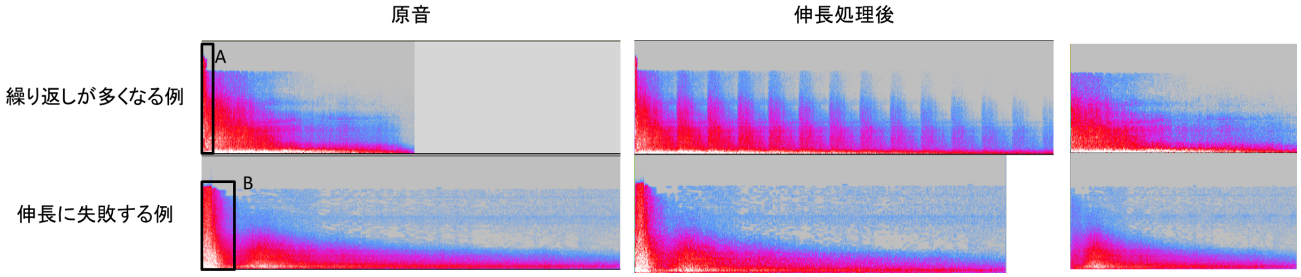

図 11 伸長に失敗した爆発音のスペクトログラムの例

8. おわりに

本研究では, 持続性のある効果音を劣化を抑えて伸長するアルゴリズムを考案した. さらにドラッグ＆ドロップのみで効果音の貼り付け作業を行うインタフェースを実装し, 効果音の編集に不慣れなユーザでも容易かつ効率的に無音動画に効果音を貼り付けることができるシステムを開発した. 考案したアルゴリズムでは効果音の音量が特に大きくなるピーク部分に着目し, 原音のピーク部分を除去して得られる波形を原音に重ね合わせることで, 音量の極端な変化を抑えた伸長を行う. また波形の重ね合わせ処理では, 伸長後の効果音の長さのみの入力で伸長可能な手法を考案し, ユーザが直感的に効果音の伸長作業を行うことを可能にした. 評価実験では, 既存のソフトウェアにより無音動画に効果音を貼り付けた結果との比較を行い, 既存手法より劣化を抑えて持続性のある効果音の伸長が行われることを確認した. また, 既存のソフトウェアを用いた場合と比較して効果音の貼り付け作業に要する時間が大幅に短縮されることを確認した.

今後は第 7 章で述べた, 音量が急速に小さくなる効果音やピーク部分が複数回存在する効果音の伸長を可能にするよう伸長アルゴリズムを改良していく. また効果音の編集作業の経験のないユーザに実際にシステムを使用してもらい, 既存のソフトウェアを用いた場合との作業時間と伸長した効果音の質の比較を行うことにより, インタフェース面での評価とアルゴリズム面での評価を行っていく予定である.

9. 謝辞

本研究の一部は,JSPS 科研費 23500114 の助成を受けたものである.

参考文献

[1] 飯島 智恵, 岡部 誠, 尾内 理紀夫, “逆再生を利用した効果音の伸長手法”, 情報処理学会 第 73 回全国大会講演論文集 (1), pp.259-261, 2011.3.

[2] 鈴木 喜也, 岡部 誠, 尾内理紀夫, “無音動画に対する効果音貼り付けシステムの試作”, In DEIM 2012.

[3] Rabiner L.R.and Schafer R.W, “Digital Processing of Speech Signals”, 1978.

[4] Reinier W. L. Kortekaas and Armin Kohlrausch, “Psychoacoustical evaluation of the pitch-synchronous overlap-and-add speech-waveform manipulation technique using single-formant stimuli”, 1996.

[5] Werner Verhelst and Marc Roelands, “An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech”, Acoustics, Speech, and Signal Processing, 1993. ICASSP-93, 1993 IEEE International Conference on.

[6] Shahaf Grofit and Yizhar Lavner, “Time-Scale Modification of Audio Signals Using Enhanced WSOLA With Management of Transients”, IEEE Transactions on Audio, Speech, and Language Processing Volume 16 Issue 1, January 2008.