Music Mosaic Generator: 高精度時系列メタデータを利用した音楽リミックスシステム

Music Mosaic Generator: Music Remix System Using High Precision Time-series Metadata

宮島 靖*

Summary. 音楽知識を持たない一般ユーザであっても，簡単に音楽リミックス制作ができるシステム “Music Mosaic Generator” を開発した．楽曲にあらかじめ付与した高精度時系列メタデータを利用し，自動的に複数の楽曲のビートやキーを揃えることで，音楽知識や面倒な波形編集を使わずに複数楽曲を簡単に組み合わせることが可能となった．メタデータの付与には，同時に開発したメタデータオーサリングシステム “MetaPong!” を使用する．制作したリミックスは，リミックス情報を記述したレシピと原曲音源によってリアルタイムに再生成されるため，第三者とレシピだけを交換することで合法的に既存曲を使用したリミックス交換が可能となる．本研究により，楽曲を受動的に聴くだけだったユーザが能動的に音楽制作に関わる，新しい音楽 CGM 文化の創造が期待できる．

1 はじめに

一般ユーザが制作したコンテンツのことを CGM (Consumer Generated Media) と呼ぶ．写真や動画の CGM は急速に一般的なものとなったが，音楽に関しては CGM が普及しているとは言い難い．音楽を制作するためには，専門知識と楽器／ツールを使いこなす能力が要求されるため，一般ユーザはプロフェッショナルが制作したコンテンツをそのまま聴いて楽しむことが未だに主流となっている．一方で，一から作曲を行うのではなく，既存の楽曲や音源を巧みに組み合わせ，楽曲同士を繋いだりアレンジをしてしまうリミックスという手法は，主に DJ(Disc Jockey) によって盛んに使われ，PC 上で動作するリミックス指向の音楽制作ソフトウェアも発売されている [1][2].

このような既存の音源を利用したリミックス制作は，一からの作曲に比べ，次の点で CGM に向いている．

- 作曲に比べると敷居が低い
- 既存の楽曲を使うためクオリティが確保される
- 聴く側にとっても既知の楽曲が別のアレンジに変化する意外性が楽しい

上記の理由により，リミックスは今後の音楽 CGM の主流になる可能性がある．ところが，リミックス制作もこのままでは一般には普及しないと考えられる．既存の音楽制作ソフトウェアによるリミックス制作は，作曲に比較して敷居が低いとは言え，最低限の音楽知識や忍耐強い波形編集作業を行わなくてはならず，一般ユーザから見ると依然として障壁が高い．DJ にとっても波形編集は創造性とは無関係の大変面倒な作業である．どの曲とどの曲を合わせるとよいリミックスになるかは実際にやってみないと分からず，波形編集を使った Try and Error の繰り返しは創造性の発揮とは程遠いルーチンワークでしかない．また，既存曲をリミックスし再録音したものを合法的に共有する仕組みが存在しないため，インターネットを利用して広く聴いてもらえる機会がない．

本稿では，上記の問題点を克服し，一般ユーザでも簡単にリミックス制作／共有ができるシステム “Music Mosaic Generator”（以下 MMG）を提案する．

2 方針と方法論

リミックス制作における問題点を克服するために，MMG では次の方針を掲げた．

- 音声波形を見せないユーザインタフェース
- 編集作業におけるルーチンワークの自動化
- リミックス可能な楽曲の推薦
- レシピ（リミックス情報）交換による合法的なリミックス共有

2.1 リミックスの種類

一言でリミックスと言ってもさまざまな手法が存在する．もっとも大掛かりな手法は原曲の譜面をもとにしながら，演奏する楽器やメロディそのものに変更を加える方法である．これはスタジオでの新たな楽器演奏と再録音が必要になる．次に原曲の構成を入れ替えたり，他の曲や音源を重ねてアレンジを

* Yasushi Miyajima 株式会社ソニーコンピュータサイエンス研究所 インタラクションラボラトリー

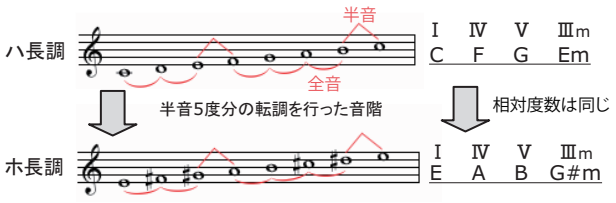

図 1. コードと度数

加える方法がある．最後に曲の繋ぎ方に着目し，テンポを調整しビートを合わせながら次の曲にシームレスに繋いだり，テンポを保ったままカットインで繋ぐなどの手法がある [5]．現在の DJ が主に行っているのは後者二つであり，MMG においても後者二つのリミックス方法を当初の対象とすることにした．

2.2 方法論

現代のポップスやロックなどは基本的に拍子リズムを刻むビートとコード進行から構成される．ビートは小節の頭のビートとそれ以外に分類でき，拍子によって小節内のビートの数と拍の分解能（四分音符，八分音符など）が決定する．なお本稿では，ビートは 1 拍をあらわす単位として定義している．すなわち，4/4 拍子の楽曲では 1 小節は 4 つのビートで構成される．コード進行はその楽曲のキー (主音) およびスケール（長調，短調など）をベースとした和音列であり，相対的な度数（ローマ数字の I,II,…VII）で表される．同じスケールで同じ相対度数の進行をもつコード列は，たとえキーが違ったとしても転調を行えば絶対的なコード進行が同じになるという特徴を持つ (図 1)．現代音楽はキー，スケール，コードの理論に従って作られており，それらの理論はさまざまな書籍によって明文化されている [10][11]．すなわち，ある楽曲におけるビート位置，キーとスケール，コード進行の情報が分かっていれば，異なる楽曲同士でビートを同期させたり必要な度数だけ転調することが自動的にできることになる．そこで，楽曲にあらかじめビート位置やコード進行などの時系列メタデータを付与しておき，その情報をもとに自動的に複数の楽曲のビート位置合わせと適切な転調を行えばよいと考えた．また，メタデータを利用することで音声波形を見せないユーザインタフェースの実現や，リミックスに適した楽曲部分の検索機能の実現が可能になると考えた．

さらに，リミックス情報を保持するレシピと原曲データからリミックスをリアルタイムに再生成する方法をとることで，レシピだけを他人と交換する合法的なリミックス交換が可能になると考えた．

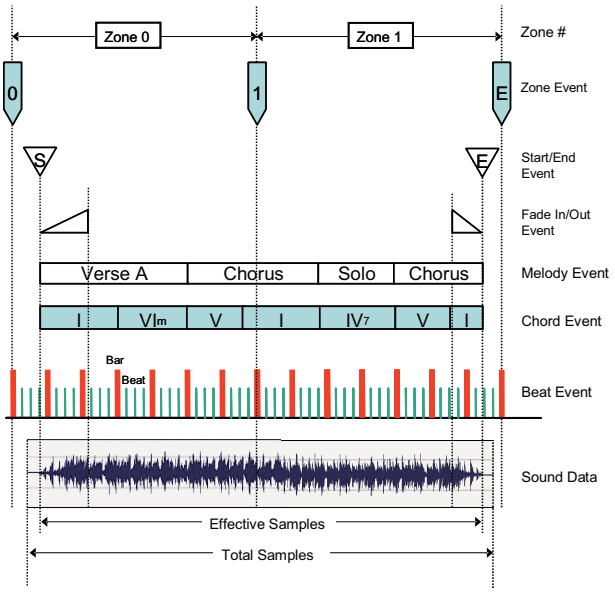

図 2. 時系列メタデータの構造

3 時系列メタデータ

まず，MMG を実現するために必要な時系列メタデータ（以下メタデータ）の種類と構造を定め，同時にそれらを付与するためのオーサリングシステムの開発を行った．MMG の詳細を述べる前に，メタデータについての要点を述べておく．

3.1 メタデータの構造

図 2 にメタデータの構造を示した．メタデータ列における個々のデータをイベントと呼び，次に示す種類が定義済みである．

ゾーンイベント 楽曲内で同じキー，スケール，拍子が続く区間を区切ったものをゾーンと呼び，その区切り境界でゾーンイベントが打たれている．転調のある曲や変拍子の曲ではゾーンが複数になる．

ビートイベント 楽曲のビート位置に打たれているイベントである．小節の頭のビートイベントには小節の頭を示すフラグが立っており，他のビートと区別ができるようになっている．例えば 4/4 拍子の楽曲の場合には，4 分音符 4 拍毎に小節フラグが立ったビートイベントが付与される．

コードイベント コードの変化点に打たれているイベントである．コードイベントは根音（ルートノート）のキーからの相対度数，m7（マイナーセブンス）や sus4（サスペンデッドフォー）などの派生コード番号で表される．

メロディ構造イベント イントロ，A メロディ，サ

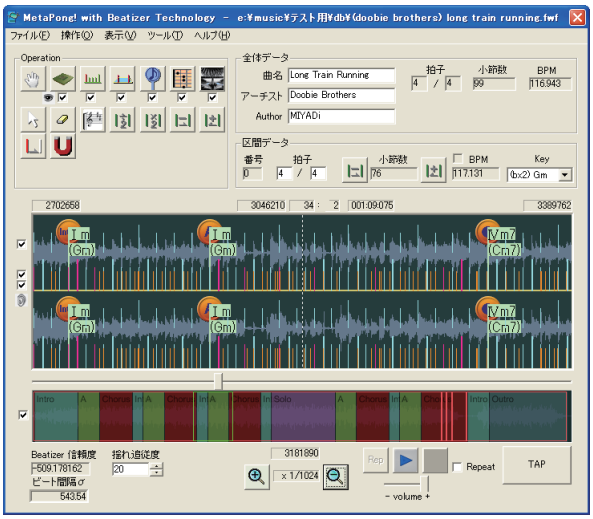

図 3. メタデータオーサリングシステム “MetaPong!”

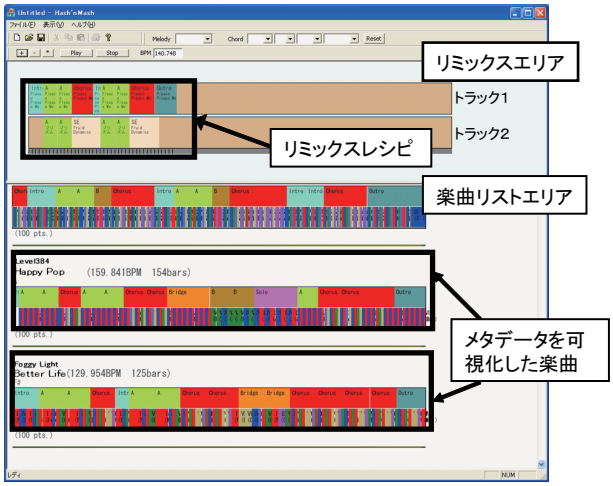

図 4. MMG アプリケーション画面

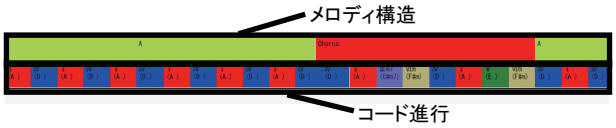

図 5. メタデータの可視化

ビなどのメロディ構造の変化点に打たれるイベントである．

その他　上記以外にもフェードイン，アウトの範囲や有効なサンプル範囲を示すイベントが存在する．

上記のイベントは，イベントの種類を表す 8 ビットのカテゴリ番号，付随するフラグやパラメータを表す 24 ビットの詳細情報，原曲データにおけるサンプリング位置を示す 32 ビットのサンプル位置情報で構成されている．

また，イベントデータとは別に，補助データとして各ゾーンの属性を示すゾーン情報ブロックが存在し，ゾーン毎のキー，スケール，拍子，テンポ情報を保持している．

3.2　オーサリングシステム “MetaPong!”

MMG で利用するビート位置やキー，コード情報などのメタデータは高精度で正確なものが要求される．これらを間違えると，ビートを同期するためのタイムストレッチ量や転調するためのピッチシフト量を正しく計算することができなくなり，結果として出力されるリミックスは聴くに耐えない品質になってしまう．しかしながら，現在の技術ではさまざまな楽曲に対して MMG が必要とする精度のメタデータを自動で付与することはできない．そのため，信号処理による自動検出と人間による修正操作によって正確なメタデータを付与するためのオーサリングシステム “MetaPong!” を開発した (図 3)．

MetaPong!によるメタデータ付与は一般ユーザが行うのではなく，ある程度音楽的知識を持った人物が付与し，多数のユーザがそのメタデータを暗黙のうちに利用することを想定している．本稿で述べているメタデータは客観的な情報であり，一度正確なメタデータを誰かが付与すれば，多数のユーザはそのままそれを利用することができる．

4　ユーザインタフェース

図 4 は MMG アプリケーション画面である．上段はリミックスエリアと呼び，2 つのトラックがあり，ここに楽曲の一部分（ブロックと呼ぶ）を配置していくことでリミックスの制作を行う．トラックの横軸は時間軸であり，右に行くほど時間が経過する．複数のトラックは同時に発音を行うため，トラック同士の発音は混ざり合うことになる．トラックの数は論理的な上限はないが，プロトタイプでは最低限の 2 トラックを用意した．下段エリアは楽曲リストエリアと呼び，楽曲素材がメタデータを元に可視化された状態で列挙される．このように，一切音声波形を見せずにブロックを組み合わせていくユーザインタフェースを実現した．

4.1　メタデータの可視化と操作

図 5 に示したように，1 つ 1 つの楽曲はメロディ構造とコード進行のイベントを可視化して表示する．上段半分がメロディ構造を表しており，同じ種類のメロディは同じ色で表示される．下段半分はコード進行を表しており，相対度数表示（I,II,…）と絶対音表示 (C,D,E,…) の両方が表示される．コードの根音度数が同じであれば同じ色で表示される．

メタデータを可視化することで，従来波形編集で行われていたようなサンプル単位の編集から脱却し，

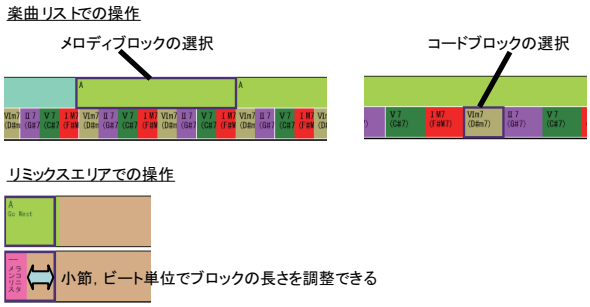

図 6. ブロック操作

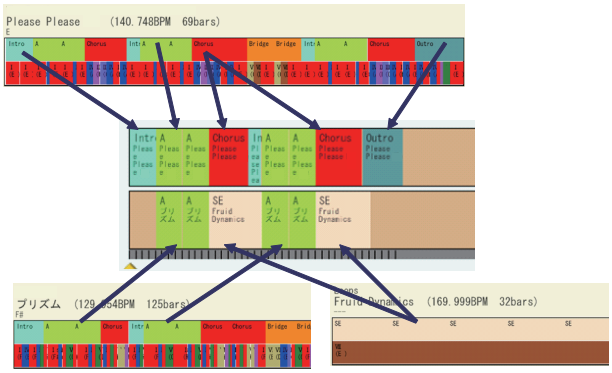

図 7. レシピの編集作業

より高次の意味的単位で操作をすることができる．例えば，図 6 に示したように，直接メロディ構造やコード単位でブロック選択が可能になる．また，ブロックの長さ（原曲から利用する範囲）の調整も小節やビート単位での調整が可能となる．これらブロックの選択，配置，範囲指定を組み合わせるだけで図 7 に示したような複雑なリミックスレシピを容易に完成させることができる．

4.2 コード進行を利用した楽曲部分推薦

リミックスの前段階において，ある楽曲と合う他の楽曲や音素材を選ぶために，数ある楽曲同士のスケールとコードの関係や拍子を確認する作業は音楽的知識と膨大な時間が必要となる．MMG ではメタデータを利用することで，この部分を自動化した．例えば，ある楽曲の一部分であるブロック A をリミックストラックに配置すると別のトラックに “Click to Recommend” という枠が出現する．この枠をクリックすることで，ブロック A のコード進行と同じコード進行部分を持つ楽曲を検索し，結果をハイライトして列挙する (図 8)．この機能があるため制作の前段階における楽曲選択を強力に支援することが可能となる．

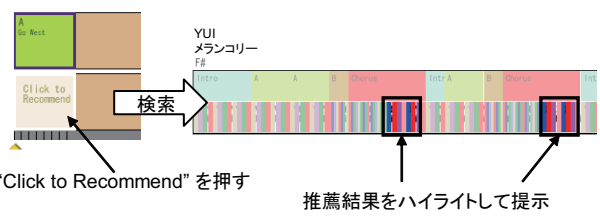

図 8. 楽曲の部分推薦

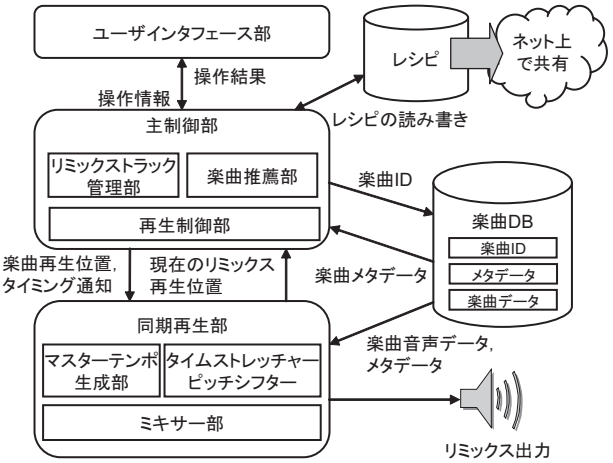

図 9. ソフトウェア構成

5 設計と実装

5.1 ソフトウェア構成

図 9 に MMG のソフトウェア構成を示した．楽曲データベースは楽曲の ID，音声データ，メタデータを保持している．楽曲の ID は 16 バイトからなり，その値は原曲音声データの一部分から暫定的に生成しているが，最終的には楽曲をユニークに特定できる ID 生成技術と融合する必要がある．主制御部はレシピを読み込み，そこに記述されているブロックをすべてトラックに展開し，使用されている楽曲 ID をキーにしてデータベースに問い合わせ，その実行環境における楽曲実体の保存場所とメタデータを特定する．主制御部と同期再生部は連携しながら後述する信号処理をリアルタイムに行う．

5.2 レシピの構造

例として 2 つのリミックストラックにそれぞれ 4 つずつのブロックを持つ場合のレシピのデータ構造を図 10 に示した．この図のトラック 2 のように，同一トラック上の各ブロックは隣接していなくてもよく，ブロックが存在しない隙間の部分は無音となる．レシピのデータ構造は，そのレシピに使用されているトラック数と各トラックデータ列からなり，各トラックデータ列はトラック内のブロック数と各ブロックデータ列から構成される．

原曲のサンプル開始点 Ss と終了点 Se は，必ず小節頭ビートの位置を示すようになっている．つま

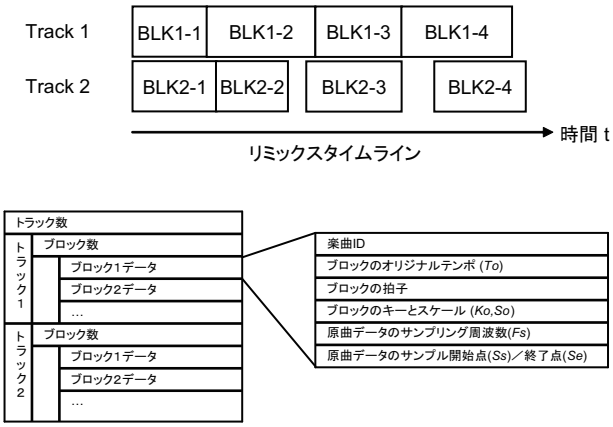

図 10. レシピのデータ構造

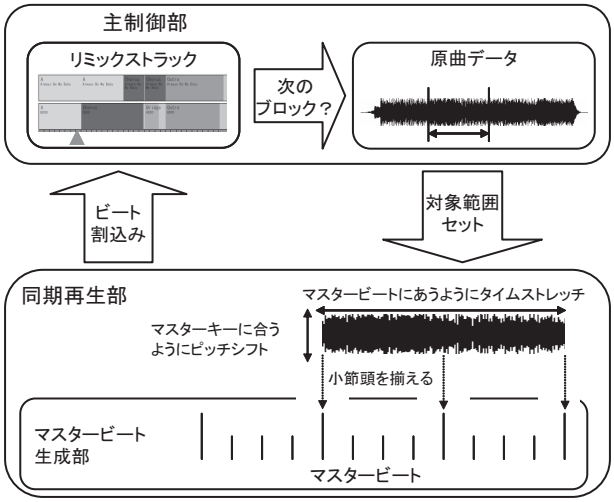

図 11. 信号処理概要

	I	IIm	IIIm	IV	V	VIm	VIIm
I	10		5			5	
IIm		10		5			5
IIIm	5		10		5		
IV		5		10		5	
V			5		10		5
VIm	5			5		10	
VIIm		5			5		10

空欄は 0

図 12. コード類似度マトリックス（長調）

り，原曲の小節頭ではない中途半端なサンプル位置は指定できない．オリジナルテンポ To は，Ss と Se が示す範囲における原曲のオリジナルテンポを示す実数値であり，BPM(Beats Per Minute) という音楽のテンポを示す単位で表す．ブロックは原曲の一部分を切り出したものであり，その範囲によってテンポが異なる場合があるため，曲全体の平均テンポを使わずにブロック毎のテンポを計算して求めている．To は次式によって求められる．

$$To = \frac{\text{ビート数} \times Fs}{Se - Ss} \times 60 \qquad (1)$$

ビート数はサンプル区間 Ss と Se におけるビート数をメタデータからカウントして求める．このとき，サンプル位置 Se のビートはカウントに含まない．

キー Ko は 1(C),2(C#),...12(B) までの整数値で表し，スケール Ks は 1 が長調,2〜4 が短調を表す．短調が 3 種類あるのは自然的，和声的，旋律的の 3 種類が存在するためである．これらキーとスケールの値はメタデータより取得する．

5.3 リアルタイム信号処理

図 11 はリアルタイム信号処理の概略図である．主制御部はリミックストラックに展開されたレシピと現在の再生位置を比較して，次のブロックの開始点の 1 小節手前まで到達すると，次に再生されるべきブロックデータを同期再生部に渡す．同期再生部はマスタービート生成部が生成する次の小節頭ビートが来た瞬間に，渡されたブロックデータに記述されている楽曲音声データの Ss サンプル目から再生を開始する．このとき，マスターテンポ Tm とブロックのオリジナルテンポ To の比率 Tm/To 倍の再生速度に設定しタイムストレッチを行う．同様にマスターキー Km とブロックのキー Ko から度数差の絶対値 d を求め，周波数比を $(\sqrt[12]{2})^d$ 倍にピッチシフトして再生する．この信号処理を各トラック毎に行う．なお，MMG で使用しているタイムストレッチとピッチシフトは再生速度と音程を独立して制御できる信号処理モジュールを使用している．

再生制御部はマスタービートが 1 つ進むたびに主制御部に対してビート割込みを発生する．主制御部はその割込みをカウントしてリミックストラック全体の再生位置を進め，上記の処理を繰り返す．

5.4 楽曲部分推薦アルゴリズム

あるブロックのコード進行と同じコード進行を持つ他の楽曲の部分を検索する機能を実装した．例えば 4/4 拍子の楽曲において V のコードが 1 小節続く場合，数値列を 5,5,5,5 とする．Vm のようにマイナーコードの場合にはマイナーであることを識別するために 16 進数の 80h を加えた値 85h を数値とする．このように，あるビート位置におけるコードを 1 バイトの数値で表し，その数値列を比較して同じコード進行部分を検索する．厳密に同じコードを探すためには sus4 などの派生部分も含めて比較するべきであるが，敢えて派生部分は検索に使用していない．これは，派生部分が異なったコードを組み合わせることで，偶然生まれる響きの可能性を残すためである．

さらに，あいまい検索の機能も実装した．音楽理論では「代理コード」と呼ぶ，あるコードの代わりに使っても響きに違和感がない別のコードが知られている．例えばハ長調におけるIのコードの構成音はC,E,Gであり，IIImの構成音はE,G,Bである．この2つのコードを比較するとE,Gの2音が同じ音であり，Iの代わりにIIImのコードを使用しても響きに違和感がない．このような代理コードの性質を利用して図12に示したマトリックスを使い，コード同士の類似度に点数をつけ，それらの総和を用いて，あるコード列と別のコード列の類似度を算出している．なお，コード類似度マトリックスは長調／短調では異なる．

6 関連研究

楽曲の音声信号に対し信号処理によりメタデータを抽出する技術が研究されている．ビートの自動抽出はドラム音などのはっきりしたビート音が含まれていない録音に対してもビートを抽出する方法が考案されている [8][6]．コード進行抽出やメロディ構造抽出に関しても研究がなされている [4][7]．これらの手法では，現在のところ100%正確なデータを自動抽出できていないが，MetaPong!と併用することで将来より簡単に正確なメタデータを付与できる可能性がある．

ネット上で複数の人間が音楽制作／共有を行う方法の研究が行われている．Splice[3] およびCC-Remix[9] は，ネット上で複数の人たちがリミックス制作を行うために開発されたソフトウェアである．これらは，自分で演奏した音源やCreative Commonsのライセンスを利用した音源を共有するため著作権問題が起こらないが，市販楽曲は利用できない．また，両者とも利用者の敷居を下げる工夫はしているが，メタデータを利用したMMGに比べるとシステムが自動で行ってくれる支援は限られている．

7 まとめと今後

MMGによって従来の煩わしい波形編集から脱却し，より短時間でより簡単にリミックス制作ができるようになった．また，ユーザ同士でレシピのみを交換し，レシピと各自が持っている原曲からリアルタイムにリミックスが生成される方式としたことで，合法的なリミックス交換が可能となった．具体的な評価は今後の課題であるが，参考までに実例を挙げておくと，デモ用に制作した4分程のリミックスは1時間以内で完成させることができた．従来の方法では経験上10時間以上かかると思われる．MMGにより，音楽CGM文化創造のための一つの方法論を示せたと考えている．より実用的なリミックスシステムにするためには，トラック数の増加，フィルタ，エフェクトなどの実装が必要である．

大きな問題点としては，正確なメタデータの付与を前もって行わなければならない点である．この作業をユーザ各自がそれぞれに行うのでは効率が悪い．この点に関しては，一度誰かが正確なメタデータを付与し，インターネットのサーバ上で蓄積／共有することが望ましいと考える．

また，蓄積したメタデータはMMGのみならず，音楽と同期したスライドショーアプリケーション，楽器アプリケーション，音楽解析などの応用が考えられる．まずは研究用途に本メタデータフォーマットを公開し，広く利用してもらえる環境を整備したいと考えている．

謝辞

MetaPong!におけるビート自動抽出部の実装では株式会社ソニー・コンピュータエンタテインメントアーキテクチャ研究部の山下功誠氏のご協力をいただいた．タイムストレッチおよびピッチシフトの信号処理部分はソニー株式会社 技術開発本部 信号処理技術部の話速変換ライブラリを利用させていただいた．

参考文献

[1] ACID. http://www.sonycreativesoftware.com/.

[2] Live! http://www.ableton.com/.

[3] Splice. http://www.splicemusic.com/.

[4] A.Sheh and D.P.Ellis. Recognition Using EM-Trained Hidden Markov Models. In *Proceedings of the International Symposium on Music Information Retrieval*, pp. 185–191, 2003.

[5] F. Broughton and B. Brewster. *How to Dj Right: The Art and Science of Playing Records*. Grove Pr, 2003.

[6] G. Masataka. An Audio-based Real-time Beat Tracking System for Music With or Without Drum-sounds. In *Journal of New Music Research Vol.30 No.2*, pp. 159–171, 2001.

[7] M.Cooper and J.Foote. Automatic Music Summarization via Similarity Analysis. In *Proceedings of the International Symposium on Music Information Retrieval*, pp. 81–85, 2002.

[8] M.E.P.Davies and M.D.Plumbley. Beat Tracking With A Two State Model. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. Vol.3, 241–244, 2005.

[9] A. Tanaka, N. Tokui, and A. Momeni. Facilitating collective musical creativity. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pp. 191–198, 2005.

[10] 篠田元一. 実践コード・ワーク 理論編. リットーミュージック, 2005.

[11] 石桁 真礼生, 末吉保雄他. 楽典—理論と実習 (新装版). 音楽之友社, 2001.