音源分離を用いた持ち寄り Android 端末による 3D モデルライブ

杉田 陽亮[1,a)]　荒生 太一[1]　成見 哲[1,b)]

概要：混合音源から各楽器音源を取り出す複数音源分離技術は，コンピュータの性能向上によって性能的にも速度的にもある程度満足出来るレベルに達してきている．ここでは日々性能向上しているスマートフォンの GPU 機能を活用することにより，音源分離処理を Android 端末上で行う．モバイルであることを生かしたデモとして，それぞれの楽器に合わせて 3D キャラが動くことで，好きな音楽のライブをその場で再現するシステムを提案する．

1. はじめに

複数の要素から構成される混合音源が与えられたとき，元々の要素に分離する処理を音源分離という．音源分離は多くの手法が提案され，分離精度，処理速度の両面で良い性能を実現してきている．

また，スマートフォンは人々に広く普及し多くの場面で利用されている．スマートフォンには計算処理を行う System on a Chip(SoC) と呼ばれるチップが搭載されており，日々性能が向上している．SoC には基本的な演算を行う CPU に加えて，3D 映像などを高速に処理する Graphics Processing Unit(GPU) が集約されている．GPU は並列処理を得意としており，この高い計算能力を様々な用途で利用するものを General Purpose computing on GPU(GPGPU) という．

本研究では Android 端末に組み込まれた GPU を利用し，高い計算能力を駆使することで音源分離を端末上で実装する．サーバー等を経由することなく端末上で処理を完結させることで，余分な通信が発生せず携帯性も向上する．

一方，近年の Virtual Reality (VR) 技術の発展に伴い，現実空間と仮想空間にまたがるような音楽体験の場が生まれてきている．石山らは現実のドラムと仮想空間のドラムの位置を合わせ，遠隔地のプレーヤーと合奏を行えるシステムを提案している [1]．Cluster と呼ばれる VR サービス内では音楽ライブも行われている [2]．本研究ではモバイル端末を持ち寄った状況を想定し，お気に入りのバンド音楽をその場でお気に入りの仮想バンドに演奏させる仕組みを提供する．サーバーとなる Android 端末上でバンド音楽に対し音源分離を行い，それぞれのクライアント端末上で対応する音源の楽器を 3D キャラに演奏させる．端末の位置をユーザが自由に決めることで，元音源とは異なる配置の別のリアルなバンド音楽の体験が期待できる．

[1] 電気通信大学
[a)] s1831081@edu.cc.uec.ac.jp
[b)] narumi@cs.uec.ac.jp

2. 音源分離

音源分離とは，様々な要素が混合した音源から元の音源を分離する処理である．一般的には音声処理も対象とするが，本研究では音楽分野，特に CD や配信で用いられるモノラル・ステレオ信号かつボーカル・ギター・ドラムなど複数楽器音源に対する音源分離に着目する．

複数音源分離を実現する手法は数多く提案されているが，音響処理的手法と Deep Neural Network(DNN) を利用した手法に大別できる．

2.1 音響処理的手法

音響処理的な手法として，非負値行列因子分解 (Non-negative Matrix Factorization; NMF) を利用するものが挙げられる．NMF は式 (1) にて表される．

$$X \approx TV \tag{1}$$

ここで X は混合音源のスペクトログラム，T は音色を表す基底，V は時間の特徴を表すアクティベーションである．具体的な手法として，NMF に追加の情報を与えず基底のクラスタリングを適用する手法 [3]，音色の情報を学習する教師あり手法 [4] などが挙げられる．

2.2 DNN を利用した手法

DNN を複数音源分離に適用する手法は，前処理にスペ

表 1　実験利用端末

	Huawei MediaPad T2 8.0 Pro
SoC	Snapdragon 615
CPU	64bit 8core 最大 1.7GHz
GPU	Adreno405 550MHz
RAM	2GB

表 2　計測結果

Window size	CPU [s]	GPU [s]	Speed up rate
128	68.427	6.440	10.63
256	63.900	6.303	10.14
512	63.325	6.155	10.29
1024	56.600	5.995	9.44

クトログラム化を行うもの，前処理を行わず音源データをそのまま扱うものが存在する．前者の例としては MM-DenseLSTM[5]，後者の例としては Wave-U-Net[6] 等が挙げられる．

スペクトログラムを扱うものは画像処理と同様のアプローチが行えることから多くの手法が提案されているが，変換途中の短時間フーリエ変換 (Short-Time Fourier Transform; STFT) で情報欠落が生じる．反対に音源をそのまま扱うものはその情報欠落が生じないが，近年提案されたものであり現在は精度面で課題が残っている．

3. Android における実装

3.1 Android 端末上での GPGPU

Android 端末上で GPGPU を行う手法として，様々な環境で並列計算を実現するフレームワークである OpenCL[7] を利用した．Android 上での OpenCL API の呼び出しは，Java Native Interface(JNI) を経由し C 言語や C++にて記述することで可能となる．また，あらかじめビルド対象端末から OpenCL 共有ライブラリ (libOpenCL.so) を抽出し，アプリに内包することでプログラムを実行できる．

3.2 テスト計算の性能評価

Android 端末における CPU と GPU の性能比較として，STFT とその逆変換である inverse STFT(iSTFT) の処理時間を用いた．実験端末は**表 1** に記載した．

評価には長さ 3 分 20 秒，サンプリング周波数 44.1kHz のモノラル音源を使用した．STFT は窓幅 $N = \{128, 256, 512, 1024\}$ とし，シフト幅は窓枠の半分とした．音源に STFT と iSTFT を連続して適用し，元の音源に戻るまでの処理時間を計測した．

STFT は窓幅毎に繰り返し高速フーリエ変換 (FFT) を適用する処理であり，この繰り返しの並列化を検討する．CPU はマルチスレッドを利用した並列処理，GPU は 2 次元空間を考え，x 軸方向に窓幅，y 軸方向に各 FFT 処理を行う並列処理とした．この結果を**表 2** に記載した．

どの窓幅においても約 10 倍の高速化を確認し，STFT に対して GPU 処理が有効だといえる．Ross らが N 体問題のベンチマークにおいて 10 倍程度の加速をしていることから，妥当な速度を実現していると考えられる [8].

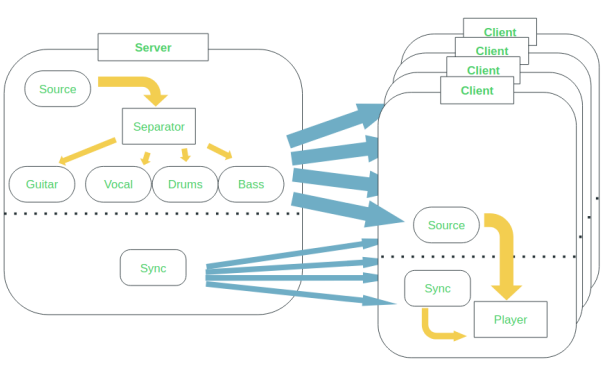

図 1　ライブシステム概要図

3.3 Android 端末上での音源分離

2 節で述べた種々の手法について Android 端末への実装の方針について述べる．

NMF を利用した手法は行列演算を主としており，STFT と同様に実装することで高速化が見込める．ただし，精度と計算速度を考慮した基底サイズや NMF の収束条件の決定が困難である．

DNN を利用したものは深層学習ライブラリ TensorFlow のモバイル端末版である TensorFlow Lite の利用によって実現できる．あらかじめ PC 上で構築・学習したモデルをモバイル向けに変換し，変換後モデルをアプリに搭載することで端末上で処理が可能となる．しかし，全ての処理をモバイル向けに変換できないこと，深いネットワークの推論に足る処理性能がないことから課題が残る．既存のモデルに対して，精度や処理速度を考慮した変更を加えることでこの課題の解決を図る．

4. 3D モデルライブシステム

4.1 システム概要

提案する 3D モデルライブシステムの概要を**図 1** にて示す．サーバー端末において前述した音源分離をバンド音源に対して行い，ボーカル・ギター・ベース・ドラム等それぞれのパート音源を得る．得られた音源を各クライアント端末へ送信する．クライアント端末では Unity を用いてパートに応じた楽器を 3D モデルに演奏させる．音源の送受信は端末間でローカルなネットワークを構築することで，どんな環境でも端末を持ち寄りさえすればシステムの動作が可能となる．ローカルなネットワークの構築には，Wi-Fi 機能による PeerToPeer 通信を実現する Wi-Fi Direct を利用する．

図 2 3D モデル演奏デモ

4.2 3D モデルによる演奏

3D モデルライブの動作の一例が**図 2** である．1 端末が 1 楽器を演奏することで，元の混合音源では一意に定まっていた音源位置をユーザが自由に決めることができる．また，3D モデルによる演奏を交えることで視覚的なライブ感も得られる．

Unity のアセットである FinalIK を利用し，3D モデルに楽器演奏を模したモーションを行わせた．また，音量に対して閾値を定め，再生音源に対して一定音量を越した場合にのみモーションさせることで音楽とのタイミングを合わせ演奏感の向上を図った．

4.3 端末間の同期

音源は端末間で同期を取り，同期情報を元に再生する．現在の方針では Network Time Protocol(NTP) の利用を考えている．あらかじめクライアント端末の時刻を NTP 基準で設定する．そしてサーバー端末からある時刻を指定した再生開始情報を配信する．クライアント端末は同期情報を受け取り，指定時刻に再生を開始することで同時刻に演奏を開始できる．

より高精度な同期機構が必要であれば、4K や 8K 放送用のプロトコルである MPEG Media Transport(MMT) などの使用も検討する [9].

5. おわりに

音源分離を Android 端末上で実装し，これを利用した 3D モデルによるライブシステムを提案した．今後は音源分離の種々の手法の検討と，3D モデルの演奏クオリティの向上による演奏感の向上が課題である．

参考文献

[1] 俊之石山，鉄朗北原：HMD を用いたヴァーチャルなドラム演奏環境の試作：合奏相手を表すヴァーチャルキャラクターの導入，エンタテインメントコンピューティングシンポジウム 2018 論文集，pp. 76–79 (2018).

[2] cluster: cluster(クラスター) | バーチャルイベント空間，Cluster Inc.（オンライン），入手先 ⟨https://cluster.mu/⟩（参照 2019-07-29）.

[3] Kameoka, H., Nakano, M., Ochiai, K., Imoto, Y., Kashino, K. and Sagayama, S.: Constrained and regularized variants of non-negative matrix factorization incorporating music-specific constraints, *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5365–5368 (2012).

[4] Kitamura, D., Saruwatari, H., Shikano, K., Kondo, K. and Takahashi, Y.: Music signal separation by supervised nonnegative matrix factorization with basis deformation, *2013 18th International Conference on Digital Signal Processing (DSP)*, pp. 1–6 (2013).

[5] Takahashi, N., Goswami, N. and Mitsufuji, Y.: Mmdenselstm: An Efficient Combination of Convolutional and Recurrent Neural Networks for Audio Source Separation, *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, pp. 106–110 (2018).

[6] Stoller, D., Ewert, S. and Dixon, S.: Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation, *Proceedings of the 19th International Society for Music Information Retrieval Conference*, pp. 334–340 (2018).

[7] KhronosGroup: OpenCL Overview, The Khronos Group Inc (online), available from ⟨https://www.khronos.org/opencl/⟩ (accessed 2019-07-05).

[8] Ross, J. A., Richie, D. A., Park, S. J., Shires, D. R. and Pollock, L. L.: A case study of OpenCL on an Android mobile GPU, *2014 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–6 (2014).

[9] 侑輝河村，一博大槻，洋介遠藤：MMT を用いた端末間映像同期の Android 端末への実装，エンタテインメントコンピューティングシンポジウム 2016 論文集，pp. 85–92 (2016).