

リサンプルブロック処理と並列化に基づく ICA の実時間実装

武田龍[†] 中臺一博[‡] 高橋徹[†] 尾形哲也[†] 奥乃博[†]

[†]京都大学大学院情報学研究科 [‡](株)ホンダ・リサーチ・インスティテュート・ジャパン

Real-time Implementation of ICA based on Resample-based Block-wise and Parallel Processing

*Ryu Takeda[†], Kazuhiro Nakadai[‡], Toru Takahashi[†], Tetsuya Ogata[†], and Hiroshi G. Okuno[†]

[†]Graduate School of Informatics, Kyoto University

[‡]Honda Research Institute Japan Co., Ltd.

Abstract— This paper describes a speed-up and performance improvement of multi-channel semi-blind ICA (MCSB-ICA). MCSB-ICA is an integrated method of sound source separation that accomplishes blind source separation, blind dereverberation, and echo cancellation. The main problem when MCSB-ICA is applied to robot audition is its high computational cost. We tackle this by multi-threading programming, and the two main issues are 1) the design of parallel processing and 2) incremental implementation. These are solved by a) multiple-stack-based parallel implementation, and b) resampling-based overlaps and block-wise separation. The experimental results proved that our method reduced the real-time factor to less than 0.5 with an eight-core CPU, and it improves the performance of automatic speech recognition by 2-10 compared with simple parallel-implementation.

Key Words: Robot audition, ICA, real-time processing, blind source separation, blind dereverberation, echo cancellation

1. はじめに

ロボットが実環境で自律的な音声インタラクションを行うには、音響的な次の3つの課題、1) 複数音の聞き分け、2) バージン（割り込み）発話、3) 部屋の残響、への対応が必須である（図1）。ロボット聴覚研究や音響信号処理分野では、これら複数の課題を扱っている研究はまだ少なく、挑戦的な課題の一つである。

我々は、独立成分分析 (ICA) を応用することで、これらの問題を統一的に解決する手法: Multi-channel Semi-blind ICA (MCSB-ICA) を開発してきた。一般的なICAと違い、残響時間の線形オーダの演算量で、残響環境下でも十分な分離性能を発揮できる [1]。一方、ロボットへ搭載する際に課題となる、リアルタイム処理に関しては扱われていない。MCSB-ICA は残響を分離するために時系列情報を利用しており、通常のICAで用いられている手法を適用できない [2, 3]。

本稿では、MCSB-ICA のリアルタイム処理を実現するために、a) 並列プログラミング実装、b) リサンプルに基づくオーバーラップブロック処理、を導入する。これらにより、演算量を抑えた実時間動作、かつ分離性能の低下を抑えた処理が可能となる。

2. MCSB-ICA による音源分離

本章では、後半の並列処理の説明に必要最小限のMCSB-ICA の分離アルゴリズムを説明する。分離の前処理と後処理の説明は省くため、詳しくは文献 [1] を参考にされたい。

2.1 変数の定義

MCSB-ICA のモデルは短時間フーリエ変換 (STFT) 領域 [4] で記述される。周波数ビン $\omega \in \{N, N -$

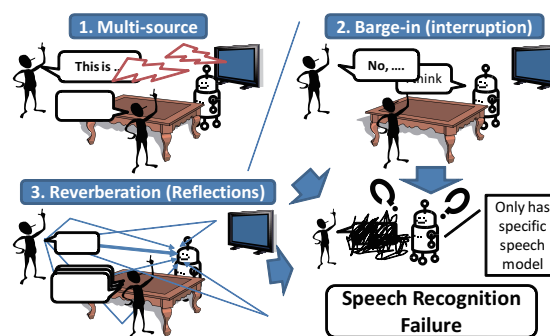


Fig.1 実環境における音響的問題

$1, \dots, 0\}$, フレーム番号 t における STFT された信号のスペクトルを $s_\omega[t]$ と表す。すべての分離処理は各周波数ビン毎に行われることに注意されたい。

マイクロホン M_1, \dots, M_L (L はマイクロホン数) で観測されたスペクトルをそれぞれ $z_{\omega,1}[t], \dots, z_{\omega,L}[t]$ で記述し、そのベクトル形式を $z_\omega[t] = [z_{\omega,1}[t], \dots, z_{\omega,L}[t]]^T$ で表す。既知であるロボット発話のスペクトルを $\tilde{s}_{r,\omega}[t]$ とする。MCSB-ICA の入力観測スペクトル $z_\omega[t]$ とロボット発話スペクトル $\tilde{s}_{r,\omega}[t]$ であり、出力は推定されたユーザ発話スペクトル $\hat{s}_\omega[t]$ である。

ここで、観測ブロックベクトル $Z_{d,\omega}[t]$ 、及び、既知信号ベクトル $S_{r,\omega}[t]$ を、初期間隔パラメータ d 及び、フィルタタップ M_{bd} と M_{ec} を用いて次のように定義する。

$$Z_{d,\omega}[t] = [z_\omega[t-d], \dots, z_\omega[t-d-M_{bd}]]^T, \quad (1)$$

$$S_{r,\omega}[t] = [\tilde{s}_{r,\omega}[t], \dots, \tilde{s}_{r,\omega}[t-M_{ec}]]^T. \quad (2)$$

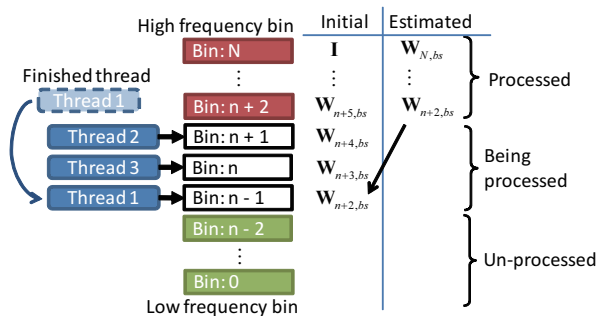


Fig.2 SSI の挙動

2.2 分離フィルタ推定

分離信号ベクトル $\hat{s}_\omega[t]$ は、次の分離モデルに従って、観測スペクトルから獲得される。

$$\hat{s}_\omega[t] = \mathbf{W}_{bs,\omega} \mathbf{z}_\omega[t] + \mathbf{W}_{bd,\omega} \mathbf{Z}_{d,\omega}[t] + \mathbf{W}_{ec,\omega} \mathbf{S}_{r,\omega}[t] \quad (3)$$

ここで、 $\mathbf{W}_{bs,\omega}$ 、 $\mathbf{W}_{bd,\omega}$ と $\mathbf{W}_{ec,\omega}$ は、それぞれブラインド分離、ブラインド残響除去、エコーキャンセルの分離行列に対応する。 \hat{s}_ω の次元は L であり、 $\mathbf{W}_{bs,\omega}$ 、 $\mathbf{W}_{bd,\omega}$ 及び $\mathbf{W}_{ec,\omega}$ のサイズは、それぞれ $L \times L$ 、 $L \times L(M_{bd}+1)$ 、 $L \times (M_{ec}+1)$ である。

このモデルは行列形式に変換できるので、ICA の枠組みで分離行列を推定できる。 \mathbf{W}_{bs} 、 \mathbf{W}_{bd} と \mathbf{W}_{ec} の学習則は以下の通りである。

$$\mathbf{W}_{bs,\omega}^{[j+1]} = \mathbf{W}_{bs,\omega}^{[j]} + \mu \left[\mathbf{D}_\omega \mathbf{W}_{bs,\omega}^{[j]} \right], \quad (4)$$

$$\mathbf{W}_{bd,\omega}^{[j+1]} = \mathbf{W}_{bd,\omega}^{[j]} + \mu \left[\mathbf{D}_\omega \mathbf{W}_{bd,\omega}^{[j]} - \mathbb{E}[\phi(\hat{s}_\omega[t]) \mathbf{Z}_{d,\omega}^H[t]] \right], \quad (5)$$

$$\mathbf{W}_{ec,\omega}^{[j+1]} = \mathbf{W}_{ec,\omega}^{[j]} + \mu \left[\mathbf{D}_\omega \mathbf{W}_{ec,\omega}^{[j]} - \mathbb{E}[\phi(\hat{s}_\omega[t]) \mathbf{S}_{r,\omega}^H[t]] \right], \quad (6)$$

$$\mathbf{D}_\omega = \text{diag}[\mathbb{E}[\phi(\hat{s}_\omega[t]) \hat{s}_\omega^H[t]]] - \mathbb{E}[\phi(\hat{s}_\omega[t]) \hat{s}_\omega^H[t]] \quad (7)$$

ここで、 $[j]$ は反復回数、 μ はステップサイズパラメータ、 $\phi(x) = [\phi(x_1), \dots, \phi(x_L)]^H$ は非線形関数ベクトルである。非線形関数には、連続領域 $|x| > \varepsilon$ で定義される $x^*/|x|$ を用いる。

2.3 MCSB-ICA の特性

周波数ビン ω における分離行列 $\mathbf{W}_{bs,\omega}^{[0]}$ の初期値は、 $\omega+1$ で推定された行列 $\mathbf{W}_{bs,\omega+1}$ を用いる。最初の分離行列の初期値は単位行列を用いる。経験的に、すべての分離行列の初期値に単位行列を用いた場合、MCSB-ICA の分離パフォーマンスは大きく低下する。

3. 並列処理実装

この章では、マルチスレッドプログラミングを用いた MCSB-ICA の並列処理に関して説明する。ここでは、 P 個のスレッドを用いた並列処理を仮定する。

3.1 MCSB-ICA の並列性

分離行列の初期値依存性を除き、周波数ビン単位での並列化が MCSB-ICA において最も効果的である。なぜなら、すべての分離は独立して行われるからである。もちろん、行列演算レベルの並列化も考えられるが、実際に実装したところ、スレッド化コストの方が大きかったため、逆に速度低下を招いた。

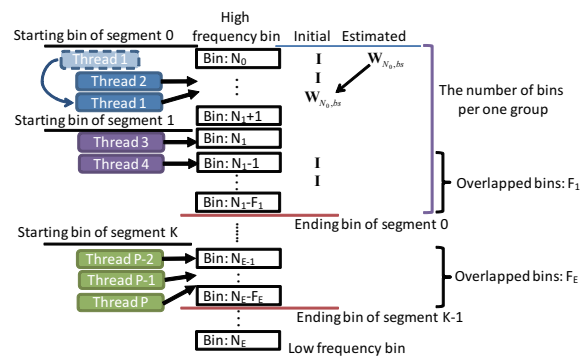


Fig.3 MSI の挙動

3.2 Single-stack-based Implementation (SSI)

単純な実装方法では、高周波数ビンから低周波数ビンに向かって処理する。分離行列の初期値は分離が終了した一番近くのビンのものを利用する(図2)。この機構は、スタックを用いて簡単に実装でき、また、スレッドも待ち状態を作ることがないので効率的である。処理手順を Alg. (1) に示す。“selectW(\mathbb{W} , x)” は、推

Algorithm 1 SSI ; apply to all threads

```

- 未処理の周波数ビン を低周波から UPBIN = {N, N-1, ..., 0} に PUSH;
-  $\mathbb{W} = \phi$ ;
while ( $\omega = \text{pop}(\text{UPBIN})$ ) != null do
-  $\mathbf{W}_{bs,\omega}^{[0]} = \text{selectW}(\mathbb{W}, \omega)$ ;
-  $\hat{s}_\omega$ ,  $\mathbf{W}_{bs,\omega}$ ,  $\mathbf{W}_{bd,\omega}$ , と  $\mathbf{W}_{ec,\omega}$  を推定;
-  $\mathbb{W} \leftarrow \mathbb{W} \cup \mathbf{W}_{bs,\omega}$ ;
end while

```

定された分離行列の集合から、最も近くかつ x より大きな値の周波数ビンで推定された $\mathbf{W}_{bs,x}$ を返す。また、もし $\mathbb{W} = \phi$ の場合や \mathbb{W} に対応するものがなければ、単位行列 I を返す。

この実装は、分離行列の初期値を隣接したものを利用する保証がないので、分離行列の初期値依存性による分離性能低下を考慮していない。

3.3 Multiple-stack-based Implementation (MSI)

分離行列初期値問題を回避するため、周波数ビンをいくつかのセグメントに分割し、それぞれのセグメントに割り当てるスレッドの数を制限する(図3)。一つのセグメント $x \in \{0, \dots, K\}$ は、 P_x のスレッドと $N_x - (N_{x+1} - F)$ 個のビンを持っている。ここで、 $N_x = N - \sum_{k=1}^x P_k / P$ であり、 F はオーバーラップしたビンの数である。このオーバーラップは分離行列の初期値問題を緩和する効果がある。我々は、スレッドを割り当てるとき、高周波ビンのセグメントは高々2スレッドしか割り当てない戦略をとり、他のところは2~3個のスレッドを割り当てる。従って、 K は P と N に従って変わる。

処理手順を Alg. 2 に示す。SSI と MSI の違いは、オーバーラップ処理があるかないかである。

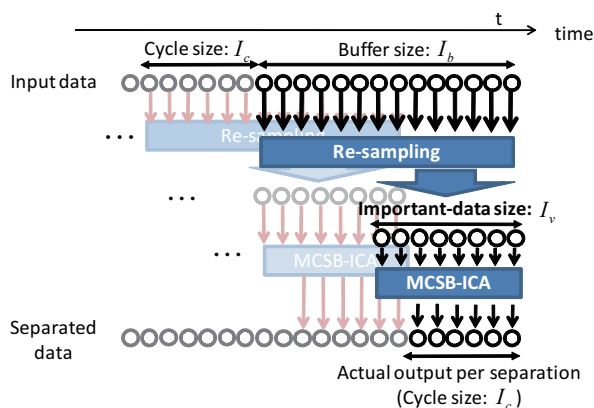


Fig.4 オーバラップ・ブロック処理

Algorithm 2 MSI for segment x

- 未処理の周波数ピンを低周波から $UPBIN_x = \{N_x, \dots, \max(N_{x+1} - F, 0)\}$ に PUSH;
 - $W = \phi$ とし、これはすべてのセグメントで共有;
 - while** ($\omega = \text{pop}(UPBIN_x) \neq \text{null}$) **do**
 - $W_{bs,\omega}^{[0]} = \text{select } W(W, \omega)$;
 - $\hat{s}_\omega, W_{bs,\omega}, W_{bd,\omega}$ と $W_{ec,\omega}$ を推定;
 - $W \leftarrow W \cup W_{bs,\omega}$;
 - end while**
-

4. オーバラップ・ブロック処理とリサンプルに基づく逐次実装

本章では、MCSB-ICA の逐次実装を説明する。MCS-ICA は、残響除去フィルタ W_{bd} を推定する時に、フレーム毎に変化する音声信号の統計的時間構造を利用しているため、従来とは異なる実装方式が必要となる。

4.1 オーバラップ・ブロック処理とリサンプルフェーズ

我々は、オーバーラップ・ブロック処理実装を採用し、オンライン分離を達成する。オーバーラップさせる理由は、十分な分離性能を発揮するには多くのサンプルが必要だからである。具体的には、時刻 $[t - I_b, t]$ のデータを分離するのに、過去 I_b までのサンプルを用いる。バッファによる遅延を減らすために、cycle size I_c を定義する。図(4)に示されるように、もし I_c が大きければ、演算量が減る代わりに遅延が増加し、 I_c が小さければ、遅延が低下する代わりに演算量が増加する。ここには2種類の遅延、バッファタイム I_b と処理時間が存在することに注意。全体の遅延はそれらを足して得られる。

演算量と遅延時間のバランスを取るために、リサンプルフェーズをブロック処理に導入する。このリサンプル処理によって、バッファされたデータが分離行列推定に用いられる Important-data size I_v にまで減らされる。もし、最適にサンプルを選ぶことができれば、低演算量処理かつ高パフォーマンスな分離が期待できる。なぜなら、分離行列推定は、式(4) - (6)に見られるように、期待値演算に基づいているからである。

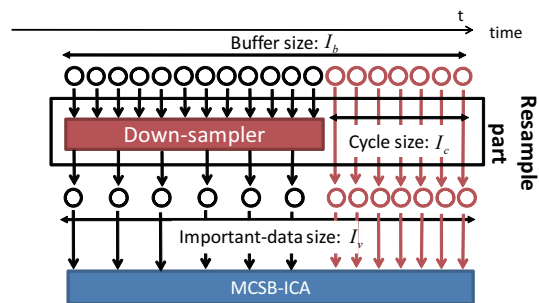


Fig.5 ダウンサンプラに基づくリサンプル実装

4.2 ダウンサンプラに基づくリサンプル実装

我々は、実験的に cycle-size のデータに対して全域通過フィルタを、オーバーラップするデータに対してはダウンサンプラを用いる(図5)。ダウンサンプラは $(I_v - I_c)$ 分のデータを対応するフレーム時刻 \tilde{t}_i から選択する。

$$\tilde{t}_i = \frac{I_b - I_c}{I_v - I_c} i + (t - I_b), \quad i = 0, \dots, (I_v - I_c) \quad (8)$$

式(4) - (6)の期待値は、リサンプルされたデータから計算される。

もちろん、important-data size の50%以上は、オーバーラップ処理により一度分離が行われているため、リサンプルを上手くする方法はまだある。このような情報を使えば、より効果的なアルゴリズムの導出が期待できるが、ここでは詳細な議論を行わず、リサンプルそのものの効果を確認するにとどめる。

5. 実験

5.1 評価項目・基準

ロボット、話者A、話者Bの3種類の音源を用意し、以下の状況における分離性能をそれぞれ評価する。

- I. 話者A (w/o noise),
- II. 話者A + ロボット (A + R),
- III. 話者A + 話者B: (A + B),
- IV. 話者A + 話者B + ロボット: (A + B + R)

以下の2つの項目に関して、単語正解率や Real-time factor (RTF) によって評価を行う。

1. SSI と MSI の分離性能
2. リサンプルオーバーラップ処理の効果

RTF は 処理時間 P と データの時間 I を用いて、 $RTF = \frac{P}{I}$ と定義される。ブロック処理では I が I_c に相当する。

5.2 実験設定

評価用データ 8本のマイクがヒューマノイドロボットの頭に装着されており、インパルス応答は16kHz サンプリングで録音した。部屋の大きさは $4.8 \times 5.55 \times 3$ [m] であり、残響時間 RT_{20} は 940 [ms] である。話者Aはロボットの正面から1.0[m]離れた場所に位置している。話者Bはロボットから1.5[m]離れており、ロボットの正面と話者の角度は、10, 20, 30, 60, 90の5パターン試した。ロボット自身のスピーカからマイクまでのインパルス応答も同様に計測した。マイクの高さは約1.25[m]であり、他のスピーカの高さは1.4[m]

Table 1 実験データおよび分離パラメータ設定

インパルス応答	16 kHz sampling
残響時間 (RT ₂₀)	940 [ms]
話者 B の位置	10°, 20°, 30°, 60°, 90°
マイク数	8 (embedded in robot's head)
STFT analysis	Hanning: 64 [ms] and shift: 20 [ms]
入力データ	[-1.0 1.0] normalized

Table 2 音声認識設定

テストセット	200 sentences
学習セット	200 people (150 sentences each)
音響モデル	PTM-Triphone: 3-state, HMM
言語モデル	trigram, vocabulary size of 21 k
音響分析	Hanning: 32 [ms] and shift: 10 [ms]
音響特徴量	MFCC 25 dim. (12+ Δ 12+ Δ Pow)

Table 3 平均単語正解率 (%)

Pattern: [$I_c I_v I_b$], [P F]	話者 A の WC				話者 B の WC	
	w/o noise	A+R	A+B	A+B+R	A+B	A+B+R
SSI: [50 150 150], [8, -]	75.9	64.1	60.5	49.3	40.7	28.7
MSI: [50 150 150], [8, 15]	79.6	68.1	63.9	53.7	44.3	34.1

Table 4 平均単語正解率 (%) 及び RTF

Pattern: [$I_c I_v I_b$]	話者 A の WC				話者 B の WC		RTF			
	w/o noise	A+R	A+B	A+B+R	A+B	A+B+R	1core	2core	4core	8core
Pat.1: [50 100 100]	77.2	65.2	58.6	48.9	38.9	29.0	3.1	1.45	0.77	0.45
Pat.2: [50 100 150]	78.1	67.8	61.9	52.0	41.1	31.5	3.1	1.49	0.77	0.45
Pat.3: [50 150 150]	79.6	68.1	63.9	53.7	44.3	34.1	4.4	2.08	1.09	0.64

である。データセットは、JNAS データベースから 200 発話選択し、それぞれのインパルス応答を畳み込み、混合して作成した。

分離パラメータ・音声認識設定 フレーム初期間隔パラメータ d は 2 とし、残響除去フィルタおよびエコーキャンセルフィルタの長さは、同じ $M_{bd} = M_{ec} = 20$ とした。ステップサイズ適応のパラメータは Takeda *et al.* [5] と同じものである。フィルタ推定における反復数は 15 とし、実験では、voice-active section は given としている。STFT パラメータ等は表 1 に示す。

HMM ベースの Julius¹ を音声認識デコーダとして用いた。音響モデルは男女各 200 人の 150 発話を用いて学習している。他の実験条件は表 2 にまとめる。

計算機環境 実験に用いた計算機は、Intel(R) Xeon(R) CPUs (X5570 2.93 GHz) を 2 つ搭載しており、合計 8 コアある。メモリサイズは 12GB、OS は Ubuntu 9.04 (jaunty)、コンパイラは Intel(R) C++ Compiler 11.1 Professional Edition for Linux、また、計算ライブラリ Math Kernel Library (MKL)² を用いた。

5.3 実験結果・考察

表 3 に、SSI と MSI の各状況での平均単語正解率を示す。各パラメータ設定は、表の左側の部分に記している。すべての音源組み合わせにおいて、MSI の方が約 3-5 ポイント WC が改善している。これより、分離行列の初期値依存性問題は、オーバーラップして再び分離を行うことで回避されていることがわかる。

次に、リサンプル・ブロック逐次処理の結果を表 4 に示す。ブロック処理では、3 種類のパラメータ [$I_c I_v I_b$] = pat.1: [50 100 100], pat.2: [50 100 150], pat.3: [50 150 150] を設定している。50 で 1 [s] に相当する。リサンプルを導入することで、pat.1 と pat.2 は RTF が同じ程度であるが、WC が 1-3 ポイントしている。pat.3 は、バッファしているすべてのデータを用いた場合の性能であり、pat.2 の性能がこれと同じ、か

つ、RTF が pat.2 と同じであれば、理想的である。また、両方で WC が 1-3 ポイント差があるため、リサンプルにおけるサンプル選択方法に改善の余地があるといえる。RTF は 8 core で 1.0 を切っているため、実時間処理が達成されている。

6. おわりに

本稿では、実環境での音源分離手法 MCSB-ICA をロボット聴覚へ搭載する際の課題である、逐次処理の実現に取り組んだ。1) 並列処理、2) リサンプルに基づくオーバーラップブロック処理を導入し、問題の解決を図った。実験の結果、演算量を抑え実時間処理を達成し、リサンプルや MSI の効果により、認識率が 5 pts 改善することを確認した。

今後は、リサンプル方法の改善と、実際に動作しているロボットへ実装を行い、音声対話システムとの統合し、音声対話ロボットとしての評価を行う予定である。謝辞 本研究の一部は基盤研究 (S)、グローバル COE、特別研究員奨励費の支援を受けた。

参考文献

- [1] R. Takeda *et al.*, "Upper-limit evaluation of a robot audition based on ICA-BSS in multi-source, barge-in and highly reverberant conditions," in *Proc. of ICRA*, 2010, pp. 4366-4371.
- [2] H. Saruwatari *et al.*, "Two-stage blind source separation based on ICA and binary masking for real-time robot audition system," in *Proc. of IEEE/RSJ IROS*, 2005, pp. 209-214, IEEE.
- [3] K. Kondo *et al.*, "A semi-blind source separation method with a less amount of computation suitable for tiny DSP modules," in *Proc. of Interspeech*, 2009, pp. 1339-1342.
- [4] T. Nakatani *et al.*, "Blind speech dereverberation with multi-channel linear prediction based on short time fourier transform representation," in *Proc. of ICASSP*, 2008, pp. 85-88, IEEE.
- [5] R. Takeda *et al.*, "Step-size parameter adaptation of multi-channel semi-blind ICA with piecewise linear model for barge-in-able robot audition," in *Proc. of IROS*, 2009, pp. 2273-2282.

¹<http://julius.sourceforge.jp/>

²<http://software.intel.com/en-us/intel-compilers/>