The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Speedup and Performance Improvement of ICA-based Robot Audition by Parallel and Resampling-based Block-wise Processing

Ryu Takeda, Kazuhiro Nakadai, Toru Takahashi, Kazunori Komatani, Tetsuya Ogata and Hiroshi G. Okuno

Abstract— **This paper describes a speedup and performance improvement of multi-channel semi-blind ICA (MCSB-ICA) with parallel and resampling-based block-wise processing. MCSB-ICA is an integrated method of sound source separation that accomplishes blind source separation, blind dereverberation, and echo cancellation. This method enables robots to separate user's speech signals from observed signals including the robot's own speech, other speech and their reverberations without a priori information. 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 points compared with the single-stack-based parallel implementation without the resampling technique.**

I. INTRODUCTION

A. Background

Our goal is to develop a robot that can recognize a user's speech from a mixture of sounds and can interact with humans naturally through speech in various environments. For example, a robot can talk with a target user near a loud television, a person may talk to it from far away, and a user can interrupt a robot's utterance and begin speaking while the robot is speaking (called "barge-in"). Since speech is the most natural communication channel for humans, such robots are useful and can help us in many situations, such as in housekeeping or rescue tasks. To achieve such a robot-audition system, we must cope with the following three problems at the same time:

1) Multi-source (speech and other noise) signals,
2) The robot's own speech signal, and
3) Their reverberations.

These problems are caused by the microphones that are installed on robot's body, and not attached close to the user's mouth (Fig. 1). This degrades the performance of conventional automatic speech recognition (ASR) seriously because many ASRs or spoken dialogue systems work well in the laboratory but not in such noisy and reverberant environments. Additionally, robots must have **the least prior information** about the environment and should be adaptive to any environments to work even in unknown environments. Therefore, we can say that robot audition presents many challenges for researchers.

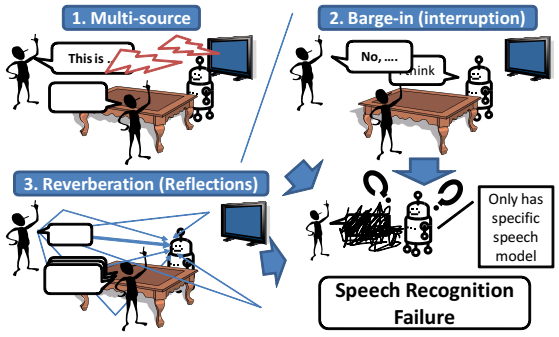

Fig. 1. Our target problems

B. Our Approach and Contribution

We adopted multi-channel semi-blind ICA (MCSB-ICA) [1] for robot audition. We used it for two reasons:

1) It is theoretically robust against Gaussian noise, such as that from fans, and
2) It can theoretically deal with blind source separation, blind dereverberation (separation of reverberation), and echo cancellation (separation of robot's speech) with the linear order calculation cost of reverberation time [2].

Here, "blind" means without a priori information.

The problem with applying MCSB-ICA to robot audition is its high computational cost, especially when many microphones are used. The number of microphones is important because the number of microphones theoretically equals the number of sound sources that can be separated.

We tackle this problem by introducing multi-threading processing to MCSB-ICA. The two main issues are 1) the design of parallel processing and 2) incremental implementation. The reason we treat the incremental implementation is that MCSB-ICA usually works with batch processing, and the techniques used by Saruwatari *et al.* [3] cannot be applied because the dereverberation mechanism is different from that in ordinary ICA. We cannot adopt the beamformer combination approach such as [4] because their scheme does not consider the reverberation problem which ordinary ICA cannot cope with. Therefore, it is necessary to re-estimate of the separation matrix for MCSB-ICA.

These two issues are solved by a) multiple-stack-based parallel implementation, and b) re-sampling-based overlaps

R. Takeda, T. Takahashi, K. Komatani, T. Ogata, and H. G. Okuno are with the Department of Intelligence Science and Technology. Graduate School of Informatics, Kyoto University, Kyoto, 606-8501, Japan. {rtakeda, tall, komatani, ogata, okuno}@kuis.kyoto-u.ac.jp

K. Nakadai is with Honda Research Institute Japan Co., Ltd., Wako, Saitama, 351-0114, Japan. nakadai@jp.honda-ri.com

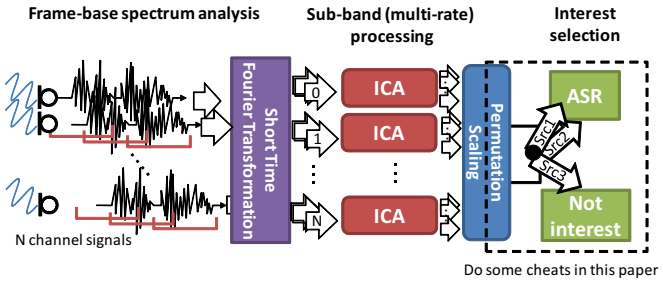

Fig. 2. Overview of whole separation process.

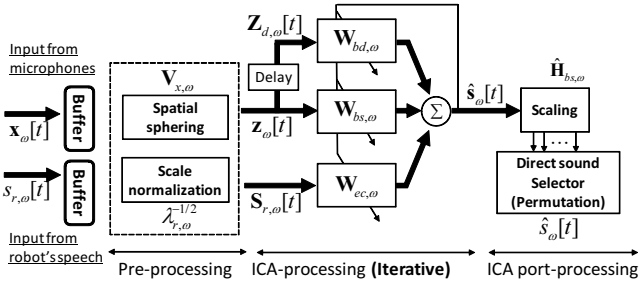

Fig. 3. Signal flow of MCSB-ICA.

and block-wise separation. The former technique is one of the designs of parallel processing of ICA, and this performs better than single stack based implementation. The latter is a technique to reduce the computational cost of re-estimating the separation matrix and block-wise processing. The key points of these techniques are overlapped processing and data selection in estimating the separation filter of ICA. Note that we have **ignored** the problems of *permutation* to evaluate the pure performance of MCSB-ICA, and that we can use the permutation solvers [5], [6] that have been proposed in other papers in practical use.

C. Organization of paper

This paper consists of seven sections. Section 2 explains the MCSB-ICA algorithm. Section 3 explains the parallel implementation, and Section 4 presents the block-wise processing with the re-sampling technique. We discuss evaluations of our method in Sections 5 and 6. The last section concludes the paper and discusses future work.

II. ALGORITHM FOR MULTI-CHANNEL SEMI-BLIND ICA

We only focus on the core algorithm of MCSB-ICA because its other parts are not as closely related to the design of parallel and block-wise processing. Please see Takeda *et al.*[1] for the precise modeling and deviation of the algorithm. The signal flow for MCSB-ICA is outlined in Fig. 3.

A. Definition of Variables

The MCSB-ICA model described here uses a short-time Fourier transformation (STFT) representation [7], which is a form of multi-rate processing (Fig. 2). We denote the spectrum after STFT as $s_\omega[t]$ at frequency-bin index $\omega \in \{N, N-1, ..., 0\}$ and frame index t. **Note that all separation process are executed in all frequency bins**.

We denote the spectra observed at microphones $M_1, ..., M_L$ as $x_{\omega,1}[t], ..., x_{\omega,L}[t]$ (L is the number of microphones) and its vector form as $\boldsymbol{x}_\omega[t] = [x_{\omega,1}[t], ..., x_{\omega,L}[t]]^T$. We also represent a known-source (robot's) spectrum as $s_{r,\omega}[t]$. The inputs of MCSB-ICA are observed spectrums $\boldsymbol{x}_\omega[t]$ and robot's spectrums $s_{r,\omega}[t]$ and the output is an estimated user's spectrum, $\hat{s}_\omega[t]$.

B. Pre-processing

Linear transformation called spatial sphering is applied to the observed spectrum. Spatial sphering de-correlates microphone input ignoring the time correlations. First, we decompose the correlation matrix of microphone inputs.

$$\mathrm{E}[\boldsymbol{x}_\omega[t]\boldsymbol{x}_\omega^H[t]] = \boldsymbol{E}_{x,\omega}\boldsymbol{\Lambda}_{x,\omega}\boldsymbol{E}_{x,\omega}^H \text{ and} \quad (1)$$

$$\mathrm{E}[s_{r,\omega}[t]\bar{s}_{r,\omega}[t]] = \lambda_{r,\omega}, \quad (2)$$

where $\mathrm{E}[\cdot]$ means the expectation about time t, $\cdot^H$ represents the conjugate transpose, $\boldsymbol{E}_{x,\omega}$ is the unitary matrix consisting of eigenvectors, $\boldsymbol{\Lambda}_{x,\omega}$ is the diagonal matrix with eigenvalues, and $\lambda_{r,\omega}$ is the variance of known (reference) signal.

With these values, we transform the input signals as

$$\boldsymbol{z}_\omega[t] = \boldsymbol{V}_{x,\omega}\boldsymbol{x}_\omega[t], \quad \boldsymbol{V}_{x,\omega} = \boldsymbol{E}_{x,\omega}\boldsymbol{\Lambda}_{x,\omega}^{-1/2}\boldsymbol{E}_{x,\omega}^H, \text{ and} \quad (3)$$

$$\tilde{s}_{r,\omega}[t] = \lambda_{r,\omega}^{-1/2} s_{r,\omega}[t]. \quad (4)$$

Then, the input signals are decorrelated and normalized and the known signals are also normalized.

Here, let us define the observed block vector, $\boldsymbol{Z}_{d,\omega}[t]$, and the known-source vector, $\boldsymbol{S}_{r,\omega}[t]$ with initial interval parameter d and filter taps M_{bd} and M_{ec}:

$$\boldsymbol{Z}_{d,\omega}[t] = [\boldsymbol{z}_\omega[t-d], ..., \boldsymbol{z}_\omega[t-d-M_{bd}]]^T \text{ and} \quad (5)$$

$$\boldsymbol{S}_{r,\omega}[t] = [\tilde{s}_{r,\omega}[t], ..., \tilde{s}_{r,\omega}[t-M_{ec}]]^T. \quad (6)$$

C. Separation and Filter Estimation

The estimated spectra, $\hat{\boldsymbol{s}}_\omega[t]$, are obtained as

$$\hat{\boldsymbol{s}}_\omega[t] = \boldsymbol{W}_{bs,\omega}\boldsymbol{z}_\omega[t] + \boldsymbol{W}_{bd,\omega}\boldsymbol{Z}_{d,\omega}[t] + \boldsymbol{W}_{ec,\omega}\boldsymbol{S}_{r,\omega}[t], \quad (7)$$

where $\boldsymbol{W}_{bs,\omega}$, $\boldsymbol{W}_{bd,\omega}$, and $\boldsymbol{W}_{ec,\omega}$ correspond to separation matrices for blind separation, blind dereverberation and echo cancellation. The dimension of $\hat{\boldsymbol{s}}_\omega$ is L, and $\boldsymbol{W}_{bs,\omega}$, $\boldsymbol{W}_{bd,\omega}$, and $\boldsymbol{W}_{ec,\omega}$ become $L \times L$, $L \times L(M_{bd}+1)$, and $L \times (M_{ec}+1)$ separation matrices, respectively.

Now, we have the following learning algorithms for $\boldsymbol{W}_{bs}$, $\boldsymbol{W}_{bd}$, and $\boldsymbol{W}_{ec}$.

$$\boldsymbol{W}_{bs,\omega}^{[j+1]} = \boldsymbol{W}_{bs,\omega}^{[j]} + \mu\left[\boldsymbol{D}_\omega \boldsymbol{W}_{bs,\omega}^{[j]}\right], \quad (8)$$

$$\boldsymbol{W}_{bd,\omega}^{[j+1]} = \boldsymbol{W}_{bd,\omega}^{[j]} + \mu\left[\boldsymbol{D}_\omega \boldsymbol{W}_{bd,\omega}^{[j]} - \mathrm{E}[\boldsymbol{\phi}(\hat{\boldsymbol{s}}_\omega[t])\boldsymbol{Z}_{d,\omega}^H[t]]\right], \quad (9)$$

$$\boldsymbol{W}_{ec,\omega}^{[j+1]} = \boldsymbol{W}_{ec,\omega}^{[j]} + \mu\left[\boldsymbol{D}_\omega \boldsymbol{W}_{ec,\omega}^{[j]} - \mathrm{E}[\boldsymbol{\phi}(\hat{\boldsymbol{s}}_\omega[t])\boldsymbol{S}_{r,\omega}^H[t]]\right], \quad (10)$$

where

$$\boldsymbol{D}_\omega = \mathrm{diag}(\mathrm{E}[\boldsymbol{\phi}(\hat{\boldsymbol{s}}_\omega[t])\hat{\boldsymbol{s}}_\omega^H[t]]) - \mathrm{E}[\boldsymbol{\phi}(\hat{\boldsymbol{s}}_\omega[t])\hat{\boldsymbol{s}}_\omega^H[t]]. \quad (11)$$

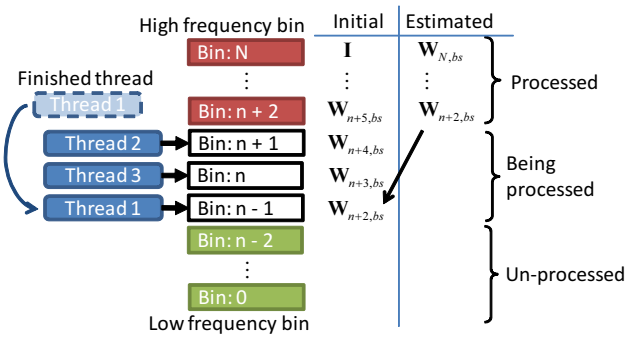

Fig. 4. Behavior based on single-stack-based implementation

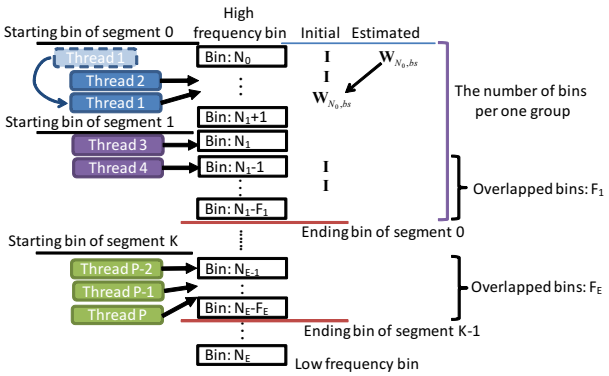

Fig. 5. Behavior based on multiple-stack-based implementation

Here, $^{[j]}$ means the number of iterations, μ is the step-size parameter, and $\boldsymbol{\phi}(\boldsymbol{x}) = [\phi(x_1), ..., \phi(x_L)]^H$ is the non-linear function vector. We used $x^*/|x|$ as a non-linear function that is defined in a continuous area, $|x| > \varepsilon$.

The initial value of the separation matrix at the frequency-bin ω, $\boldsymbol{W}^{[0]}_{bs,\omega}$, was set to that of the estimated matrix at the frequency-bin $\omega + 1$, $\boldsymbol{W}_{bs,\omega+1}$. We used the unit matrix for the initial value of the first separation matrix. Empirically, the performance of MCSB-ICA degrades if we use the unit matrix as the initial value of the separation matrix for every frequency bin.

D. Post-processing

We scale the output spectrum using the approach by Murata *et al.* [8]. We multiplied the i-th row and j-th column element c_j of $\hat{\boldsymbol{H}}_{bs,\omega} = (\boldsymbol{W}_{bs,\omega}\boldsymbol{V}_{x,\omega})^{-1}$, which satisfies Eqs. 12 and 13 for the scaling of the j-th element of $\hat{\boldsymbol{s}}_\omega[t]$.

$$l_j = \arg\max_l |\hat{\boldsymbol{H}}_{bs,\omega}(l, j))| \text{ and} \quad (12)$$

$$c_j = \hat{\boldsymbol{H}}_{bs,\omega}(l_j, j). \quad (13)$$

As mentioned in the previous section, we aligned output signals by using reference (original) signals to solve the permutation problem.

III. IMPLEMENTATION OF PARALLEL PROCESSING

This section explains the parallel processing technique for MCSB-ICA which we carry out by using multi-threading programming. We assume that P threads are available for parallel processing.

A. Parallelism of MCSB-ICA

The most efficient part for parallel processing is considered to be the separation of frequency-bin units because all separations are executed independently except for the dependence of the initial value of the separation matrix. Of course, smaller-scale parallelizations may also be effective, such as the parallel calculation of matrices. However, no matrix is too large, i.e. 160 * 300, and the cost of creating a thread is larger than that of doing calculations. This leads to increased processing time. In fact, when we use a Parallel Basic Linear Algebra subprograms (PBLAS), which is a mathematical library, the processing time becomes longer than that without parallelization.

B. Single-stack-based Implementation (SSI)

A simple implementation is processing from a high to a low frequency bin, and the initial value of the separation matrix is that of the nearest processed frequency bin (Fig. 4). This mechanism can easily be implemented by using a stack, and this implementation is efficient because no threads need to wait to begin working. This single-stack-based implementation is given in Alg. (1). Here, "selectW($\mathbb{W}$, x)"

Algorithm 1 SSI ; apply to all threads

- push unprocessed bin induces from low frequency bin to the stack, named UPBIN={N,N-1,...,0};
- $\mathbb{W} = \phi$;

while (ω = pop(UPBIN)) != null **do**

- $\boldsymbol{W}^{[0]}_{bs,\omega}$ = selectW($\mathbb{W}$, ω);
- estimate $\hat{\boldsymbol{s}}_\omega$, $\boldsymbol{W}_{bs,\omega}$, $\boldsymbol{W}_{bd,\omega}$, and $\boldsymbol{W}_{ec,\omega}$ according to Eqs. (1) – (13);
- $\mathbb{W} \leftarrow \mathbb{W} \cup \boldsymbol{W}_{bs,\omega}$;

end while

returns the estimated $\boldsymbol{W}_{bs,x}$ with the nearest and larger frequency-bin index, x, from the set of estimated matrices, $\mathbb{W}$. It returns unit matrix, I, if $\mathbb{W} = \phi$ or $\mathbb{W}$ does not include the corresponding object.

This implementation does not take into account that the influence of the initial value of the separation matrix may degrade performance because the matrix is not an adjacent one.

C. Multiple-stack-based Implementation (MSI)

We divide the frequency bins into various segments, and restrict the number of threads assigned to each segment to avoid the initial value problem (Fig. 5). A segment, $x \in \{0, ..., K\}$, has P_x threads and $N_x - (N_{x+1} - F)$ bins, where $N_x = N(N - \sum_{k=1}^{x} P_k)/P$ and F is the number of overlapping bins. This overlap has the effect of avoiding the initial value problem. In assigning threads, the highest frequency-bin segment only has two threads to process data, and we assign two or three threads to other segments. K changes according to the P and N.

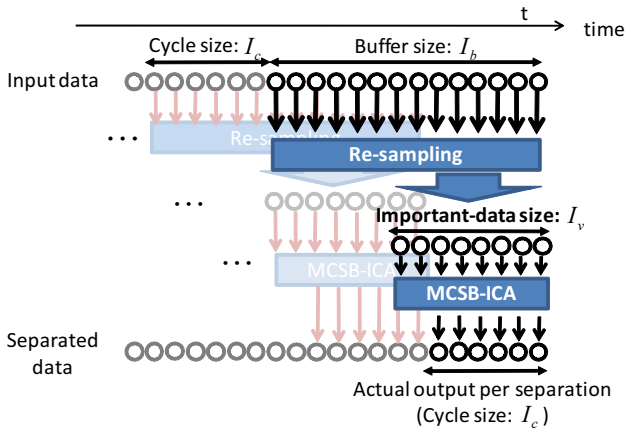

Fig. 6. Block-wise and overlapping processing

This multiple-stack-based implementation is shown in Alg. (2). The difference between SSI and MSI is that MSI has overlapping bins.

Algorithm 2 MSI for segment x

- push unprocessed bin induces from low frequency bin to the stack, named UPBIN$_x$={ N$_x$,..., max(N$_{x+1}$ − F, 0)};
- $\mathbb{W} = \phi$, and it is common object among all the segments;

while (ω = pop(UPBIN$_x$)) != null **do**

- $\boldsymbol{W}^{[0]}_{bs,\omega}$ = selectW($\mathbb{W}$, ω);
- estimate $\hat{\boldsymbol{s}}_\omega$, $\boldsymbol{W}_{bs,\omega}$, $\boldsymbol{W}_{bd,\omega}$, and $\boldsymbol{W}_{ec,\omega}$ according to Eqs. (1) – (13);
- $\mathbb{W} \leftarrow \mathbb{W} \cup \boldsymbol{W}_{bs,\omega}$;

end while

IV. BLOCK-WISE, OVERLAP AND RE-SAMPLING BASED IMPLEMENTATION

A. Block-wise implementation

We must extend MCSB-ICA to work on-line for robot audition because MCSB-ICA usually estimates the separation matrices by batch processing. Theoretically, MCSB-ICA can output separated signals frame-by-frame. However, its separation performance results in very poor outcomes and its convergence speed is critically slow because the estimation of the dereverberation matrix, $\boldsymbol{W}_{bd}$, uses the statistical time structure of the speech signal and its properties vary from frame-to-frame.

We adopted block-wise implementation for on-line processing. Since ICA buffers various duration data to stably estimate the separation matrix, we use the previous I_b samples (buffer-size) for time $[t - I_b \quad t]$ separation. We introduce cycle-size I_c to reduce latency caused by buffering. As illustrated in Figure 6, if I_c is increased, the latency also increases instead of low computational cost; if I_c is reduced, computational cost increases instead of achieving low latency. **Note that there are two kinds of delay; the first is buffering time I_c and the second is processing time**. The total delay is obtained by adding them.

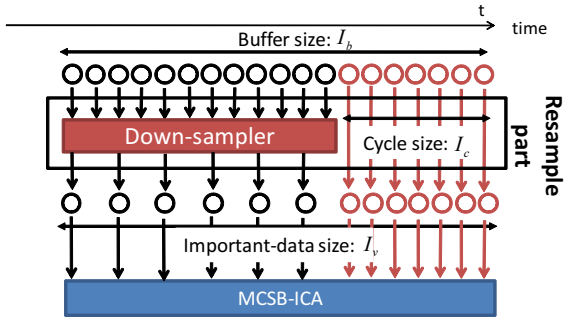

Fig. 7. Our re-sampling scheme

A re-sampling phase is introduced to block-wise processing to balance the computational cost and latency. This re-sampling reduces the number of samples to the important-data size, I_v, which is used for MCSB-ICA and selected from the buffered data. If we choose the samples optimally, high-performance and low-computational-cost processing will be achieved because filter estimation is based on the expectation operator as seen in Eqs. (8) – (10).

B. Re-sampling Technique for Overlapped Processing

We empirically adopt an all-pass filter for cycle-size data and down-sampling for overlapping data, as shown in Fig. 7. The down-sampler selects the $(I_v - I_c)$ data corresponding to the the following time index, $\tilde{t}_i$.

$$\tilde{t}_i = \frac{I_b - I_c}{I_v - I_c} i + (t - I_b), \quad i = 0, ..., (I_v - I_c) \tag{14}$$

The expectations in Eqs. (8) – (10) are calculated by using these re-sampled data.

Of course, there will be more techniques that perform well for re-sampling MCSB-ICA because we already know the separated data that make up more than 50% of the important data, which is caused by overlapped processing. A more effective algorithm may be derived if we use this information. However, we have not focused on this precisely in this discussion and have only examined the performance of re-sampling.

V. EXPERIMENTS

We evaluated MCSB-ICA by using ASR performance with simulated data. The data were generated by using the impulse responses recorded in a real environment, and impulse responses can reconstruct acoustic property, such as reflections. Note that robot's noises, such as fan noise, are not recorded and not used.

A. Experimental Settings

1) Data for evaluation: The impulse responses for speech data were recorded at 16 kHz in a reverberant room, whose RT$_{20}$ was about 940 [ms]. Here, RT$_{20}$ means the reverberation time. The size of the room was 4.8 × 5.55 × about 3 [m] (depth x width x height). The target speaker was 1.0 [m] from a microphone mounted on the head of humanoid robot. The noise speaker was located 1.5 [m] from the robot, and the angles between the noise speaker and the front of the robot

TABLE I

CONFIGURATION FOR DATA AND SEPARATION

Impulse response	16 kHz sampling
Reverberation time (RT_{20})	940 [ms]
Direction of speaker B	10°, 20°, 30°, 60°, 90°
Number of microphones	Eight (embedded in robot's head)
STFT analysis	Hanning: 64 [ms] and shift: 20 [ms]
Input wave data	[-1.0 1.0] normalized

TABLE II

CONFIGURATION FOR SPEECH RECOGNITION

Test set	200 sentences
Training set	200 people (150 sentences each)
Acoustic model	PTM-Triphone: 3-state, HMM
Language model	trigram, vocabulary size of 21 k
Speech analysis	Hanning: 32 [ms] and shift: 10 [ms]
Features	MFCC 25 dim. (12+Δ12+ΔPow)

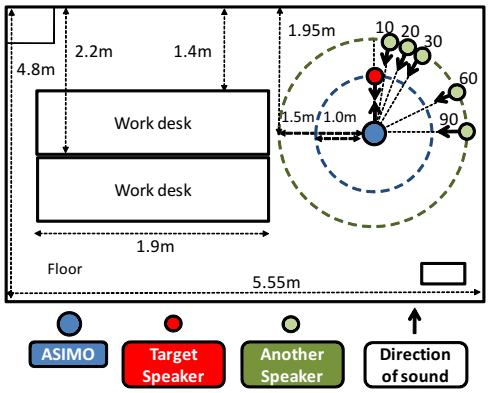

Fig. 8. Layout for room, robot and speaker.

were five patterns of 10, 20, 30, 60, 90 degrees. We also recorded the impulse response from the robot's speech. The height of the microphones was 1.25 [m] and that of the other speakers was 1.4 [m]. These settings are outlined in Figure 8. The relative amplitude of these impulse responses was saved. All data (16 bits, PCM) were normalized to $[-1.0 \quad 1.0]$ for processing.

2) Separation parameters: Eight microphones are installed on the humanoids robot developed by HONDA. The STFT parameters were set to the same values for all three experiments: the window size was 1024 points (64 [ms]) and the shift size was 320 points (20 [ms]). The frame interval parameter, d, was 2, and the default filter taps for echo cancellation and dereverberation were the same, $M_{bd} = M_{ec} = 20$. The parameters for adaptive step-size control were set to Takeda *et al.*'s settings [9]. There were 15 iterations when the filter was estimated, which is not a huge number of iterations. Note that the voice-active section was given in our experiments.

3) Configuration for ASR: We used 200 Japanese continuous speech utterances from JNAS database for the speaker's and robot's speech, and they were convoluted in the corresponding recorded impulse responses. Julius[1] was used for hidden Markov model (HMM)-based ASR with the statistical language model. Mel-frequency cepstral coefficients (MFCC) (12+Δ12+ΔPow) were obtained after STFT with a window size of 512 points and a shift size of 160 points for the speech features, and we then applied cepstral mean normalization. A triphone-based acoustic model (three-state and four-mixture) was trained with 150 sentences of clean speech uttered by 200 male and female speakers (word-closed). The statistical language model consisted of 21,000 words, which were extracted from newspapers. The other experimental conditions are summarized in Tables I and II.

4) Environments for machine and development: We used a computer with two Intel(R) Xeon(R) CPUs (X5570 2.93 GHz), and it had a total of eight cores. The memory size was 12 GB, and its operating system was Ubuntu 9.04 (jaunty) with kernel Linux 2.6-28-18-generic.

The compiler was an Intel(R) C++ Compiler 11.1 Professional Edition for Linux, and we implemented MCSB-ICA by using its Math Kernel Library (MKL) [2].

B. Combination of Noise Patterns and Evaluation Criteria

We evaluated performance of MCSB-ICA under four noise combinations:

1) Target speech,
2) Target speech and robot speech for barge-in situation,
3) Target speech and noise speech for simultaneous-talk situation, and
4) Target speech, noise speech and robot speech for worst situation.

The performances are measured by the word correctness (WC, Cor.) of the target speech (Target sp.) and noise speech (Another sp.). The correctness is defined by

$$\text{Cor.} = \frac{\text{\# of correct words}}{\text{\# of all words}}. \tag{15}$$

Cor. increases if the words in the sentence are recognized without missing words. Since insertion error can be recovered by techniques in spoken dialogue system, such as [10], we adopt WC as an evaluation criteria.

Real-time factor (RTF) is used for evaluating the processing time of MCSB-ICA. RTF is calculated by using P and I, which correspond to the process time and data time (duration), and defined by $\text{RTF} = \frac{P}{I}$. In block-wise processing, I is substituted with I_c, and then RTF is calculated.

C. Items for Evaluation

We conducted three experiments.

Exp. A: RTF of batch and other block-wise processing
Exp. B: ASR performance of SSI and MSI
Exp. C: ASR performance of HARK, batch, and other block-wise processing

First, we derive the processing speed by parallel processing in terms of the number of threads P and filter length N_{bd}. In these experiments, we used three values of $I_c = 50$ (1 [s]), $I_v = 150$ (3 [s]) and $I_b = 150$ (3 [s]). We set the overlap

[1]http://julius.sourceforge.jp/

[2]http://software.intel.com/en-us/intel-compilers/

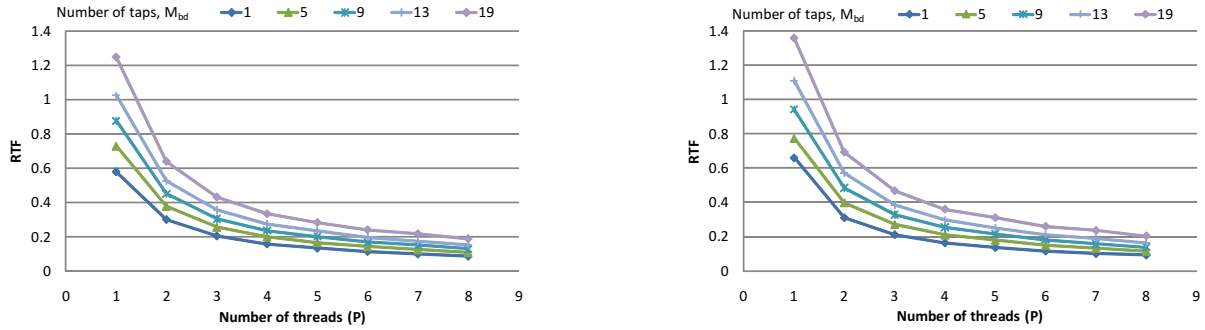

Fig. 9. RTF of batch processing. Left graph shows results without robot's own speech, and right one is with robot's speech.

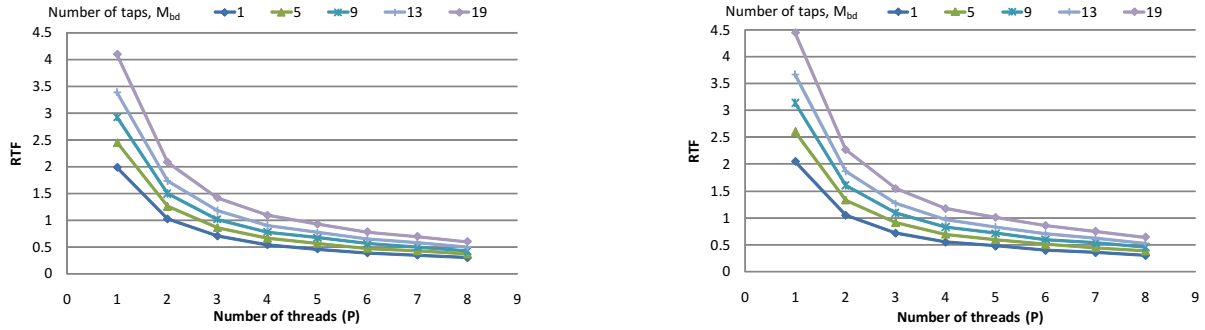

Fig. 10. RTF of block-wise processing. Left graph shows results without robot's own speech, and one at right is with robot's speech.

parameter of MSI to $F = 15(P \geq 5)$ and $F = 10(P < 5)$. Of course, batch processing uses all data for estimating the matrix.

Second, we derive the ASR performance with SSI and MSI in batch and block-wise processing. We fixed the number of threads P to eight. The parameter settings of block-wise processing were also the same as those in Exp. A.

Finally, we derive the ASR performance with re-sampling block-wise processing, and compare our method with that of other *state-of-the-art* robot audition software, HARK [11], as a baseline method. In block-wise processing, we selected three patterns for the parameters, i.e., $[I_c \ I_v \ I_b]$ = pat.1: [50 100 100], pat.2: [50 100 150] and pat.3: [50 150 150]. There are two reasons we set $I_c = 50$ and $I_b = 150$. The first for $I_b = 150$ is that MCSB-ICA needs 3 [s] of data to enable stable separation. The second for $I_c = 50$ is that 1 [s] is the maximum latency permissible by ASR. We fixed the number of threads to eight in this experiment.

We used default parameters in HARK, and did not separate the data with the robot's own speech because HARK does not have a module to remove known signals. The network file of HARK mainly consists of "GSS" and "Postfilter" module. The directions of speaker and the impulse response with 36 directions at $\mathrm{RT}_{20} = 50$ [ms] were given.

VI. DISCUSSION

A. Results

1) Exp. A: Figures 9 and 10 plot the RTFs for batch and block-wize processing. The horizontal axes represent the number of threads, and the vertical ones plot the RTFs.

In batch processing, the RTF decreases as the number of threads increases. The RTF is less than 1.0 with any filter taps and with/without the robot's own speech if we use two threads. For 19 taps, the ratio of RTF between one and eight threads becomes 6.6. The ideal ratio should be eight, and there is a 17% loss in computational efficiency. This is caused by the overlapped processing bin, F, and other processes, such as data-copy procedures.

The RTF in block-wise processing is larger than that of batch processing because it includes the overlapping separation related to the parameters, I_c and I_v. Unlike batch processing, we need six threads to satisfy RTF < 1.0. The RTF ratio between one and eight threads is 6.8 and this is almost the same as that with batch processing.

In total, we can conclude that we achieved a speedup in processing with MCSB-ICA by multi-threading programming. The total delay from the beginning of the input signal to the end of separation is about 1.5 [s] which is calculated by $I_c(1+\mathrm{RTF})$. This means that it takes only about $I_c \times \mathrm{RTF}$ [s] to achieve separation after finishing the proceeding of the buffer data.

2) Exp. B: Table III lists the ASR results for SSI and the MSI algorithm. The ASR performance with clean speech is about 92%. The "no proc." means the results without any processing. The "w/o noise" with "no proc." represents the results affected only by the reverberation of the environment.

First, block-wise processing degrades performance more than batch processing. The difference is about 15–20 points, and degradation can be distinguished for multiple sources. We can see that only dereverberation is categorized into an easy problem. This seems to be partially caused by the insufficient separation of the head of speech because there is not enough data at the beginning of speech.

Second, we focus on the results for SSI and the MSI

TABLE III

ASR PERFORMANCES (%) OF BATCH AND BLOCK-WISE PROCESSING WITH SSI AND MSI.

Noise type	angle	method	Target		Another	
			SSI	**MSI**	SSI	**MSI**
w/o noise		no proc.	26.0		16.9	
		batch	84.7	85.6	–	–
		block-wise	75.9	79.6	–	–
Speech	10	no proc.	10.9		5.0	
		batch	68.1	68.3	42.8	42.0
		block-wise	49.7	53.8	25.3	29.2
	20	no proc.	11.1		3.4	
		batch	74.2	75.8	54.9	58.0
		block-wise	57.6	62.1	37.3	42.1
	30	no proc.	10.5		3.6	
		batch	79.7	79.9	64.5	65.4
		block-wise	65.7	67.6	44.8	48.7
	60	no proc.	12.3		5.1	
		batch	80.8	81.2	68.1	67.1
		block-wise	65.7	69.3	49.5	51.5
	90	no proc.	11.7		5.5	
		batch	79.4	80.4	65.7	64.7
		block-wise	63.8	66.8	46.5	49.9
Robot		no proc.	15.9		8.4	
		batch	81.8	82.6	–	–
		block-wise	64.1	68.1	–	–
Speech & Robot	10	no proc.	9.9		3.9	
		batch	58.8	59.5	31.2	32.8
		block-wise	38.1	42.6	16.1	18.7
	20	no proc.	9.9		4.1	
		batch	68.8	69.8	46.7	50.7
		block-wise	46.7	52.0	26.2	30.9
	30	no proc.	9.6		3.8	
		batch	75.1	75.3	55.8	58.1
		block-wise	53.5	57.9	32.9	38.1
	60	no proc.	9.8		5.2	
		batch	74.7	76.1	59.0	62.0
		block-wise	53.9	57.3	34.9	41.6
	90	no proc.	9.1		5.1	
		batch	75.5	75.5	55.6	56.9
		block-wise	54.4	58.6	33.7	41.4

TABLE IV

ASR PERFORMANCES (%) AND RTF OF HARK AND BLOCK-WIZE PROCESSING.

Noise type	angle	method	Target Cor.	Another Cor.	8 threads RTF
w/o noise		HARK	18.9	–	<0.02
		batch	85.6	–	0.19
		pat.1	77.2	–	0.42
		pat.2	78.1	–	0.42
		pat.3	79.6	–	0.60
Speech	10	HARK	8.3	0.1	–
		batch	68.3	42.0	–
		pat.1	49.1	23.2	–
		pat.2	52.1	25.0	–
		pat.3	53.8	29.2	–
	20	HARK	8.5	0.1	–
		batch	75.8	58.0	–
		pat.1	55.6	36.2	–
		pat.2	59.8	38.6	–
		pat.3	62.1	42.1	–
	30	HARK	8.2	0.1	–
		batch	79.9	65.4	–
		pat.1	63.6	43.3	–
		pat.2	67.1	45.2	–
		pat.3	67.6	48.7	–
	60	HARK	9.9	0.1	–
		batch	81.2	67.1	–
		pat.1	62.7	47.8	–
		pat.2	65.2	51.5	–
		pat.3	69.3	51.5	–
	90	HARK	9.9	0.2	–
		batch	80.4	64.7	–
		pat.1	62.1	44.1	–
		pat.2	65.4	45.3	–
		pat.3	66.8	49.9	–
Robot		HARK	–	–	–
		batch	82.6	–	0.20
		pat.1	65.2	–	0.45
		pat.2	67.8	–	0.45
		pat.3	68.1	–	0.64
Speech & Robot	10	HARK	–	–	–
		batch	59.5	32.8	–
		pat.1	37.2	17.0	–
		pat.2	41.1	17.1	–
		pat.3	42.6	18.7	–
	20	HARK	–	–	–
		batch	69.9	50.7	–
		pat.1	48.1	26.4	–
		pat.2	50.2	29.5	–
		pat.3	52.0	30.9	–
	30	HARK	–	–	–
		batch	75.3	58.1	–
		pat.1	52.4	31.3	–
		pat.2	54.9	34.7	–
		pat.3	57.9	38.1	–
	60	HARK	–	–	–
		batch	76.1	62.0	–
		pat.1	53.0	35.0	–
		pat.2	57.5	38.3	–
		pat.3	57.3	41.6	–
	90	HARK	–	–	–
		batch	75.5	56.9	–
		pat.1	54.0	35.5	–
		pat.2	56.5	37.7	–
		pat.3	58.6	41.4	–

algorithm. In batch processing, there is almost no difference between SSI and MSI. In contrast, there are about 2–6 points of differences in the ASR results for block-wise processing. This suggests that the initial value of the separation matrix is very important under restricted conditions, such as the number of iterations and the amount of data for separation. These experiments revealed that MSI is more effective for multi-threading programming and ASR performance. Of course, there are some losses caused by the overlapping bin parameter, F. However, the amount of loss is only 10% of the processing time, and this is not a critical problem.

3) Exp. C: Table IV lists the ASR results and RTF with HARK and block-wise processing. Since HARK cannot handle the separation of the robot's own speech, the results for robot speech are blank. We only listed the RTF for one person because RTF only depends on the number of microphones and threads, and data size.

The results for HARK are all less than 20% for all situations, and HARK does not work at all under reverberation conditions. Of course, there are some parameters that can deal with reverberations. However, the optimum parameters differ according to the situation, and the cost of tuning is very high. In terms of computational cost, HARK does not require that many resources, and its RTF is less than 0.1 with one thread.

In terms of the effectiveness of re-sampling, we can see some improvements by comparing pat.1 and pat2. Even tough the computational costs of these two parameter settings are almost the same, the ASR results for pat. 2 are better than those for pat. 1, i.e., about 2–4 points. This means re-sampling works effectively. Since the gap between pat.2 and pat.3 is a little too large, improvements to the method of re-sampling are required.

We concluded that MCSB-ICA can work better even in block-wise processing and under a reverberant environment than HARK. We also found the advantageous effects of the re-sampling technique in this experiment.

B. Remaining Issues

Three problems remain to be solved for robot audition. The first one is the permutation solver of ICA to improve the accuracy of separation. We intend to solve this by integrating previous researches [5], [6]. The second one is an automatic identification of separated results as speech or noise to avoid unnecessary ASR processing and treat non-speech activities appropriately. It can be solved by using machine learning methods, such as GMM classification.

Last but not least, low ASR performance is caused by low separation performance when only a few samples of data are available. This is the most common problem with the statistical-separation methods. The duration of short utterances , especially, such as "Yes" , "Hi!" or "Good morning!" is usually less than 1 [s]. Since most **adaptive** statistical separation methods need at least 2 [s] of data to attain sufficient separation, these short utterances, the beginnings of long utterances or utterances of a moving talkers cannot be separated from observed signals. If we do not use adaptive methods, we must prepare large amounts of acoustic data in advance to train the separation matrix so that it should cover all the situations where the system is deployed.

This problem is difficult to solve at the level of ASR, because ASR techniques for improving performance in noisy situations, such as VAD [12] or missing-data techniques [13], assume high SNR or high-quality separation results. Since the quality of separated speech is not so good as clean speech, we must refine these methods for separated sound signals.

VII. CONCLUSIONS AND FUTURE WORK

The difficult situations for robot audition are multiple sound sources, barge-in (user's interrupting utterances during system's utterance), and reverberation or reflections. Multi-Channel Semi-Blind ICA (MCSB-ICA) separates user's speech in these situations with high performance. The critical problem in applying MCSB-ICA for robot audition is to reduce its computational cost. In this paper, we tackle the problem by multi-threading programming. For parallel processing, we develop multiple-stack-based parallel implementation and for incremental processing, we develop re-sampling-based overlapping and block-wise sound source separation. The experimental results prove that out method reduces the real-time factor to less than 0.5 with an eight-core CPU and that it improves the performance of ASR by two to ten points.

In the future, we first intend to integrate MCSB-ICA with other ASR techniques. Then, we intend to develop a total robot audition system including a spoken dialogue system that actually really works in a real environment.

ACKNOWLEDGMENTS This research was partially supported by a Grant-in-Aid for Scientific Research (S), JSPS Fellows and a Global COE Program.

REFERENCES

[1] R. Takeda *et al.*, "ICA-based efficient blind dereverberation and echo cancellation method for barge-in-able robot audition," in *Proc. of ICASSP*, 2009, pp. 3677–3680.

[2] R. Takeda *et al.*, "Upper-limit evaluation of a robot audition based on ICA-BSS in multi-source, barge-in and highly reveberant conditions," in *Proc. of ICRA*, 2010, pp. 4366–4371.

[3] 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.

[4] 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.

[5] H. Sawada *et al.*, "A robust and precise method for solving the permutation problem of frequency-domain blind source separation," *IEEE Tr. on Speech and Audio Proc.*, vol. 12, no. 5, pp. 530–538, 2004.

[6] B. Loesch *et al.*, "On the robustness of the multidimensional state coherence transform for solving the permutation problem of frequency-domain ICA," in *Proc. of ICASSP*, 2010, pp. 225–228.

[7] 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.

[8] N. Murata *et al.*, "An approach to blind source separation based on temporal structure of speech signals," in *Neurocomputing*, 2001, pp. 1–24.

[9] 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.

[10] M. Katsumaru *et al.*, "Improving speech understanding accuracy with limited training data using multiple language models and multiple understanding models," in *Proc. of Interspeech*, 2010, pp. 2735–2738.

[11] K. Nakadai *et al.*, "Design and implementation of robot audition system 'HARK' —open source software for listening to three simultaneous speakers," *Advanced Robotics*, vol. 24, pp. 739–761, 2010.

[12] J. Sohn, N.S. Kim, and W. Sung, "A statistical model-based voice activity detection," *IEEE Signal Processing Letters*, vol. 6, no. 1, pp. 1–3, 1999.

[13] M. L. Seltzer, B. Raj, and R. M. Stern, "A bayesian framework for spectrographic mask estimation for missing feature speech recognition," *Speech Comm.*, vol. 43, no. 4, pp. 379–393, 2004.