Real-Time Convolutive Blind Source Separation Based on a Broadband Approach*

Robert Aichner, Herbert Buchner, Fei Yan, and Walter Kellermann

Multimedia Communications and Signal Processing
University of Erlangen-Nuremberg
Cauerstr. 7, D-91058 Erlangen, Germany
{aichner,buchner,wk}@LNT.de

Abstract. In this paper we present an efficient real-time implementation of a broadband algorithm for blind source separation (BSS) of convolutive mixtures. A recently introduced matrix formulation allows straightforward simultaneous exploitation of nonwhiteness and nonstationarity of the source signals using second-order statistics. We examine the efficient implementation of the resulting algorithm and introduce a block-on-line update method for the demixing filters. Experimental results for moving speakers in a reverberant room show that the proposed method ensures high separation performance. Our method is implemented on a standard laptop computer and works in realtime.

1 Introduction

The problem of separating convolutive mixtures of unknown time series arises in several application domains, a prominent example being the so-called cocktail party problem, where individual speech signals should be extracted from mixtures of multiple speakers in a usually reverberant acoustic environment. Due to the reverberation, the original source signals $s_q(n)$, $q = 1, \ldots, Q$ of our separation problem are filtered by a linear multiple input and multiple output (MIMO) system before they are picked up by the sensors. BSS is solely based on the fundamental assumption of mutual statistical independence of the different source signals. In the following, we further assume that the number Q of source signals $s_q(n)$ equals the number of sensor signals $x_p(n)$, $p = 1, \ldots, P$. An M-tap mixing system is thus described by $x_p(n) = \sum_{q=1}^{P} \sum_{\kappa=0}^{M-1} h_{qp,\kappa} s_q(n-\kappa)$, where $h_{qp,\kappa}$, $\kappa = 0, \ldots, M-1$ denote the coefficients of the filter from the q-th source to the p-th sensor.

In BSS, we are interested in finding a corresponding demixing system, where the output signals are described by $y_q(n) = \sum_{p=1}^{P} \sum_{\kappa=0}^{L-1} w_{pq,\kappa} x_p(n-\kappa)$ with $q = 1, \ldots, P$. The separation is achieved by forcing the output signals y_q to be mutually statistically decoupled up to joint moments of a certain order. For convolutive mixtures, frequency-domain BSS is very popular since all techniques

* This work was partly supported by the ANITA project funded by the European Commission under contract IST-2001-34327.

C.G. Puntonet and A. Prieto (Eds.): ICA 2004, LNCS 3195, pp. 840–848, 2004.

originally developed for instantaneous BSS may be applied independently in each frequency bin. This bin-wise processing, implying a narrowband signal model is denoted here as *narrowband approach* and is described, e.g., in [6]. In the context of instantaneous BSS and narrowband approaches for convolutive BSS it is known that on real-world signals with some time-structure *second-order statistics* generates enough constraints to solve the BSS problem in principle by utilizing nonstationarity or nonwhiteness [6]. Unfortunately, this traditional narrowband approach exhibits several limitations as, e.g., circular convolution effects may arise, and the permutation problem, which is inherent in BSS, may then also appear independently in each frequency bin so that extra repair measures become necessary. In [2, 3] a class of *broadband* algorithms was derived, for both the time domain and frequency domain, i.e., the frequency bins are no longer considered to be independent for unrestricted time-domain signals. These algorithms are based on second-order statistics exploiting simultaneously nonwhiteness and nonstationarity and inherently avoid the above-mentioned problems. In this paper we present an efficient realization of one of these broadband algorithms which has led to a robust real-time implementation.

2 Generic Block Time-Domain BSS Algorithm

2.1 Matrix Formulation

To obtain a block processing broadband algorithm simultaneously exploiting nonwhiteness and nonstationarity of the source signals, it was shown in [2] that we need to introduce a block output signal matrix

$$\mathbf{Y}_q(m) = \begin{bmatrix} y_q(mL) & \cdots & y_q(mL-L+1) \\ y_q(mL+1) & \ddots & y_q(mL-L+2) \\ \vdots & \ddots & \vdots \\ y_q(mL+N-1) & \cdots & y_q(mL-L+N) \end{bmatrix}, \tag{1}$$

and reformulate the convolution as

$$\mathbf{Y}_q(m) = \sum_{p=1}^{P} \mathbf{X}_p(m)\mathbf{W}_{pq}, \tag{2}$$

with m being the block time index and N denoting the block length. The $N \times L$ matrix $\mathbf{Y}_q(m)$ incorporates L *time-lags* in the correlation matrices into the cost function defined in Sect. 2.2, which is necessary for the exploitation of the nonwhiteness property. To ensure linear convolutions for all elements of $\mathbf{Y}_q(m)$, the $N \times 2L$ matrices $\mathbf{X}_p(m)$ and $2L \times L$ matrices $\mathbf{W}_{pq}$ are given as

$$\mathbf{X}_p(m) = \begin{bmatrix} x_p(mL) & \cdots & x_p(mL-2L+1) \\ x_p(mL+1) & \ddots & x_p(mL-2L+2) \\ \vdots & \ddots & \vdots \\ x_p(mL+N-1) & \cdots & x_p(mL-2L+N) \end{bmatrix}, \tag{3}$$

$$\mathbf{W}_{pq} = \begin{bmatrix} w_{pq,0} & 0 & \cdots & 0 \\ w_{pq,1} & w_{pq,0} & \ddots & \vdots \\ \vdots & w_{pq,1} & \ddots & 0 \\ w_{pq,L-1} & \vdots & \ddots & w_{pq,0} \\ 0 & w_{pq,L-1} & \ddots & w_{pq,1} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & w_{pq,L-1} \\ 0 & \cdots & 0 & 0 \end{bmatrix}, \tag{4}$$

where the matrices $\mathbf{X}_p(m)$, $p = 1, \ldots, P$ in (2) are Toeplitz matrices due to the shift of subsequent rows by one sample each. The matrices $\mathbf{W}_{pq}$ exhibit a Sylvester structure, where each column is shifted by one sample containing the current weights $\mathbf{w}_{pq} = [w_{pq,0}, w_{pq,1}, \ldots, w_{pq,L-1}]^T$ of the MIMO filter of length L from the p-th sensor channel to the q-th output channel.

To allow a convenient notation of the algorithm combining all channels, we write (2) compactly as

$$\mathbf{Y}(m) = \mathbf{X}(m)\mathbf{W}, \tag{5}$$

with the matrices

$$\mathbf{Y}(m) = [\mathbf{Y}_1(m), \cdots, \mathbf{Y}_P(m)], \tag{6}$$

$$\mathbf{X}(m) = [\mathbf{X}_1(m), \cdots, \mathbf{X}_P(m)], \tag{7}$$

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_{11} & \cdots & \mathbf{W}_{1P} \\ \vdots & \ddots & \vdots \\ \mathbf{W}_{P1} & \cdots & \mathbf{W}_{PP} \end{bmatrix}. \tag{8}$$

2.2 Cost Function and Generic Broadband Algorithm

Based on (5) we use a cost function first introduced in [2] as a generalization of [8]:

$$\mathcal{J}(m) = \sum_{i=0}^{m} \beta(i,m) \left\{ \log\det \operatorname{bdiag} \mathbf{Y}^H(i)\mathbf{Y}(i) - \log\det \mathbf{Y}^H(i)\mathbf{Y}(i) \right\}, \tag{9}$$

where β is a weighting function with finite support that is normalized according to $\sum_{i=0}^{m} \beta(i,m) = 1$ allowing on-line or block-on-line realizations of the algorithm. For a properly chosen $\beta(i,m)$ (see Sect. 2.3) the nonstationarity of the signals is utilized for the separation. Since we use the matrix formulation (5) for calculating the short-time correlation matrices $\mathbf{Y}^H(m)\mathbf{Y}(m)$, the cost function inherently includes all L time-lags of all auto-correlations and cross-correlations of the BSS output signals. The bdiag operation on a partitioned block matrix consisting of several submatrices sets all submatrices on the off-diagonals to zero. In our case, the block matrices refer to the different signal channels and are of

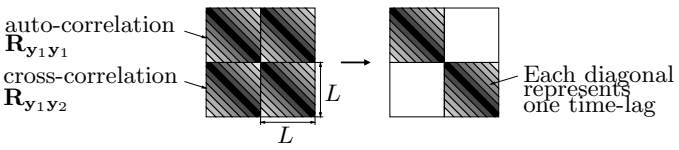

Fig. 1. Illustration of (9) for the 2×2 case

size $L \times L$. The cost function becomes zero if and only if all block-offdiagonal elements of $\mathbf{Y}^H\mathbf{Y}$, i.e., the *output cross-correlations over all time-lags*, become zero (see Fig. 1). Therefore, in addition to the nonstationarity (9) explicitly exploits the nonwhiteness property of the output signals.

In [2, 3] it was shown that the natural gradient derivation of (9) with respect to $\mathbf{W}$ leads to an iterative algorithm with the following coefficient update:

$$\nabla_{\mathbf{W}}^{\mathrm{NG}} \mathcal{J}(m) = 2 \sum_{i=0}^{m} \beta(i,m) \mathcal{Q}(i), \tag{10}$$

$$\mathcal{Q}(i) = \mathbf{W}(i) \left\{ \mathbf{R}_{\mathbf{yy}}(i) - \operatorname{bdiag} \mathbf{R}_{\mathbf{yy}}(i) \right\} \operatorname{bdiag}^{-1} \mathbf{R}_{\mathbf{yy}}(i), \tag{11}$$

where the $PL \times PL$ short-time correlation matrices $\mathbf{R}_{\mathbf{yy}}$ are consisting of the channel-wise $L \times L$ submatrices $\mathbf{R}_{\mathbf{y}_p\mathbf{y}_q}(m) = \mathbf{Y}_p^H(m)\mathbf{Y}_q(m)$.

2.3 Approximated Version and Efficient Implementation

Starting from the update equation (10) we first address implementation details concerning the update term $\mathcal{Q}(i)$ of the i-th block which are applicable regardless of the choice of the weighting function $\beta(i,m)$. In the last paragraph we specify $\beta(i,m)$ to obtain a block-on-line update rule.

Step 1: Estimation of the Correlation Matrices Using the Correlation Method. In principle, there are two basic methods to estimate the output correlation matrices $\mathbf{R}_{\mathbf{y}_p\mathbf{y}_q}(m)$ for nonstationary signals: the so-called correlation method, and the covariance method as they are known from linear prediction problems [7]. We consider here the correlation method which leads to a lower computational complexity and follows as a special case of the more accurate covariance method if we assume stationarity within each block. This leads to a Toeplitz structure of $\mathbf{R}_{\mathbf{y}_p\mathbf{y}_q}(m)$ which can be expressed as

$$\mathbf{R}_{\mathbf{y}_p\mathbf{y}_q}(m) = \left[r_{y_p y_q}(m, v-u) \right]_{L \times L} \tag{12}$$

$$r_{y_p y_q}(m, v-u) = \begin{cases} \sum_{n=mL}^{mL+N-v+u-1} y_p(n+v-u) y_q(n) & \text{for } v-u \geq 0 \\ \sum_{n=mL+|v-u|}^{mL+N-1} y_p(n+v-u) y_q(n) & \text{for } v-u < 0 \end{cases} \tag{13}$$

Step 2: Approximation of the Normalization.
A straightforward implementation of (11) together with (12), (13) leads to a complexity of $\mathcal{O}(L^2)$ due to the inversion of P auto-correlation Toeplitz matrices $\mathbf{R}_{\mathbf{y}_q\mathbf{y}_q}$ of size $L \times L$ which are normalizing the update (as also known from

the recursive least-squares (RLS) algorithm in supervised adaptive filtering [5]). Thus, for an efficient implementation suitable for reverberant environments requiring a large filter length L we use an approximated version of (11) which was first heuristically introduced in [1, 10] and theoretically derived in [2]. The efficient version is obtained by approximating the auto-correlation submatrices in the normalization term by the output signal powers, i.e.,

$$\tilde{\mathbf{R}}_{\mathbf{y}_q\mathbf{y}_q}(m) = \left(\sum_{n=mL}^{mL+N-1} y_q^2(n) \right) \mathbf{I} = \sigma_{y_q}^2(m)\mathbf{I} \tag{14}$$

for $q = 1, \ldots, P$. Thus, the matrix inversion is replaced by an element-wise division. This is comparable to the normalization in the well-known normalized least mean squares (NLMS) algorithm in supervised adaptive filtering approximating the RLS algorithm [5].

Step 3: Efficient Implementation of the Matrix-Matrix Multiplication.
In the remaining channel-wise matrix product of $\mathbf{W}_{pt}(m)$ and the Toeplitz matrices $\frac{\mathbf{R}_{\mathbf{y}_t\mathbf{y}_q}(m)}{\sigma_{y_q}^2(m)}$, $p, q, t = 1, \ldots, P$ in (11) we can exploit the Sylvester structure of $\mathbf{W}_{pt}(m)$ for an efficient implementation. Firstly, it has to be ensured that the update $\mathcal{Q}(i)$ exhibits again a channel-wise Sylvester structure in the form of (4). A simple way to impose this constraint is to calculate only the first L elements of the first column of the matrix product which contain the filter weights update $\Delta\mathbf{w}_{pq}(m)$ (see Fig. 2a). Secondly, it can be shown that this matrix product denotes a linear convolution of the filter weights $\mathbf{w}_{pt}$ with each column of $\frac{\mathbf{R}_{\mathbf{y}_t\mathbf{y}_q}(m)}{\sigma_{y_q}^2(m)}$ due to the Sylvester structure of $\mathbf{W}_{pt}$. By implementing this operation as a fast convolution using fast Fourier transforms (FFTs) the computational complexity can be reduced to $\mathcal{O}(\log L)$.

Step 4: Update Using a Block-on-Line Weighting Function
The weighting function $\beta(i, m)$ allows for different realizations of the algorithm, e.g., off-line or on-line [3]. Similar to the approach in [9] we are combining the on-line and off-line approach in a so-called block-on-line method (Fig. 2b). In Table 1 a pseudo-code for the block-on-line implementation of this efficient algorithm is given exemplarily for the filter update $\Delta\mathbf{w}_{11}$ for $P = 2$. In the block-on-line approach a block of $KL + N$ input signal samples is acquired denoted by the on-line block index m' (see Fig. 2b). K denotes the number of blocks within the offline part and thus the data is segmented into K blocks of length N with off-line block index m ($m = m' \cdot K$) and is processed by an off-line algorithm with $j_{\max}$ iterations. By simultaneously processing K blocks we exploit the nonstationarity of the signals. The implementation of the off-line part is shown in Steps 3-9 of Table 1, where j denotes the iteration number and μ_{off} is the stepsize of the off-line part.

Concerning the initialization of $\mathbf{w}_{pq}(m')$ in Step 3 for $m' = 1$ and $j = 1$, it can be shown using (4), (11) that the first coefficients of the filters $\mathbf{w}_{pp}(m')$ must be unequal to zero. Thus we use unit impulses for the first filter tap in each $\mathbf{w}_{pp}(m')$. The filters $\mathbf{w}_{pq}(m')$, $p \neq q$ are set to zero.

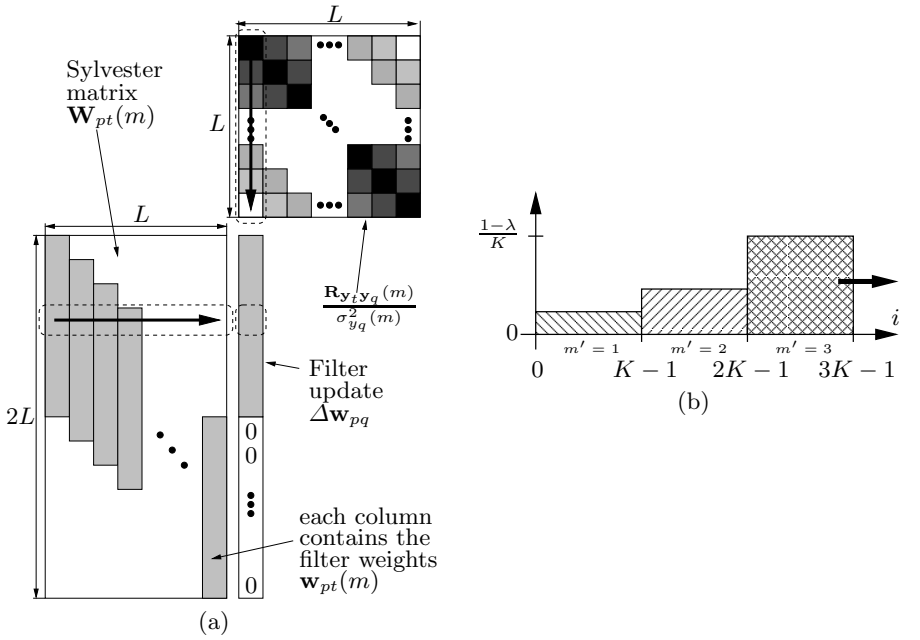

Fig. 2. (a) Illustration of the channel-wise matrix-matrix product. (b) Weighting function $\beta(i, m)$ for block-on-line implementation

The update $\Delta\tilde{\mathbf{w}}_{pq}^{j_{\max}}(m')$ (Step 9) is then used as input of the on-line part of the block-on-line algorithm. The recursive update equations of the on-line part yield the final filter weights $\mathbf{w}_{pq}$ used for separation (Step 10). Here λ denotes the exponential forgetting factor ($0 \leq \lambda < 1$) and μ_{on} is the stepsize of the on-line part. The demixing filter weights $\mathbf{w}_{pq}(m')$ of the current block m' are then used as initial values for the off-line algorithm of the next block (Step 11).

Analogously to supervised block-based adaptive filtering, the approach followed here can also be carried out with overlapping data blocks in both, the on-line and off-line part to increase the convergence rate and to reduce the signal delay. Overlapping is done by simply replacing the time index mL and $m'KL$ in the equations by $m\frac{L}{\alpha_{\text{off}}}$ and $m'\frac{KL}{\alpha_{\text{on}}}$, respectively. The overlap factors $1 \leq \alpha_{\text{off}}, \alpha_{\text{on}} \leq L$ should be chosen suitably to obtain integer values for the time index. For clarity, however, the overlap factors are omitted in Table 1.

3 Experiments and Real-Time Implementation

The experiments have been conducted using speech data convolved with the impulse responses of a real office room (580cm × 590cm × 310cm), with a reverberation time $T_{60} = 200$ ms and a sampling frequency of $f_s = 16$ kHz. A two-element microphone array with an inter-element spacing of 20 cm was used

Table 1. Pseudo code of efficient broadband algorithm implementation exemplarily shown for the update $\Delta\mathbf{w}_{11}$ in the 2×2 case

On-line part:	
1.	Acquire $KL + N$ new samples $x_p((m'-1)KL), \ldots, x_p(m'KL + N - 1)$ of the sensors x_p, $p = 1, 2$ and on-line block index $m' = 1, 2, \ldots$
2.	Generate K blocks $x_p(mL), \ldots, x_p(mL + N - 1)$ with off-line block index $m = (m'-1)K, \ldots, m'K - 1$ to enable off-line iterations
Off-line part:	
	Compute for each iteration $j = 1, \ldots, j_{\max}$:
	Compute for each block $m = (m'-1)K, \ldots, m'K - 1$:
3.	Compute output signals $y_q(mL), ..., y_q(mL + N - 1)$, $q = 1, 2$ by convolving x_p with filter weights $\mathbf{w}_{pq}^{j-1}(m')$ from previous iteration
4.	Calculate the signal energy of each block m $r_{y_1y_1}(m, 0) = \sum_{n=mL}^{mL+N-1} y_1^2(n)$
5.	Compute 1st column of cross-correlation matrix $\mathbf{R}_{\mathbf{y_2y_1}}(m)$ by $r_{y_2y_1}(m, v-u)$ for $v - u = -L + 1, \ldots, 0$ according to (13)
6.	Normalization by elementwise division $r_{y_2y_1}(m, v-u)/r_{y_1y_1}(m, 0)$ for $v - u = -L + 1, \ldots, 0$
7.	Compute the matrix product $\mathbf{W}_{12}(m)\frac{\mathbf{R}_{\mathbf{y_2y_1}}(m)}{\sigma^2_{y_1}(m)}$ as a convolution according to Fig. 2a. Each filter weight update $\Delta w_{11,\kappa}^j$, $\kappa = 0, \ldots, L-1$ is therefore calculated as: $\Delta w_{11,\kappa}^j(m') = \frac{1}{K}\sum_m \sum_{n=0}^{L-1} w_{12,n}(m) r_{y_2y_1}(m, n-\kappa)/r_{y_1y_1}(m, 0)$
8.	Update equation for the off-line part: $\mathbf{w}_{11}^j(m') = \mathbf{w}_{11}^{j-1}(m') - \mu_{\text{off}}\Delta\mathbf{w}_{11}^j(m')$
9.	Repeat Steps 3-8 for $j_{\max}$ iterations and calculate the overall update for the current m' as: $\Delta\mathbf{w}_{11}^{j_{\max}}(m') = \sum_{j=1}^{j_{\max}} \Delta\mathbf{w}_{11}^j(m')$
On-line part:	
10.	Compute the recursive update of the on-line part yielding the demixing filter $\mathbf{w}_{11}(m')$ used for separation: $\Delta\mathbf{w}_{11}(m') = \lambda\Delta\mathbf{w}_{11}(m'-1) + (1-\lambda)\Delta\mathbf{w}_{11}^{j_{\max}}(m')$ $\mathbf{w}_{11}(m') = \mathbf{w}_{11}(m'-1) - \mu_{\text{on}}\Delta\mathbf{w}_{11}(m')$
11.	Compute Steps 4-10 similarily for the other channels and use the demixing filter $\mathbf{w}_{pq}(m')$ as the initial filter for the offline part $\mathbf{w}_{pq}^0(m'+1) = \mathbf{w}_{pq}(m')$

for the recording. The speech signals arrived from two different directions, -45° and 45°. After 10 seconds one speaker position was changed from -45° to 0°. Sentences spoken by two male speakers from the TIMIT speech corpus [4] were selected as source signals. To evaluate the performance, the signal-to-interference ratio (SIR) averaged over both channels was calculated in each block which is defined as the ratio of the signal power of the target signal to the signal power from the jammer signal. Simulation results for the algorithm implemented in the real-time system are given in Fig. 3. The parameters were chosen as $L = 1024$, $N = 2048$, $K = 4$, $\alpha_{\text{on}} = 4$ resulting in a latency of 2048 samples (128 msec).

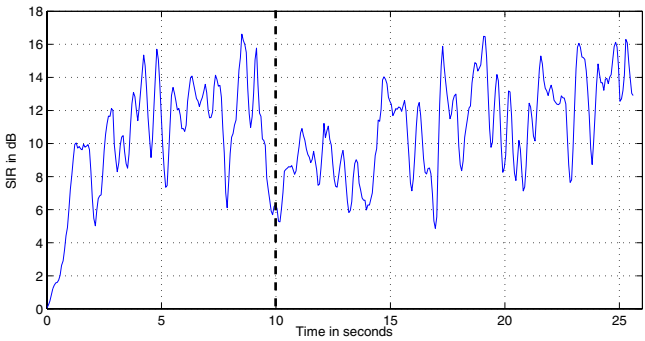

Fig. 3. Experimental results for the efficient block-on-line algorithm with an instantaneous speaker position change at 10 seconds.

The offline-part was calculated for $j_{\max} = 10$ iterations and the stepsizes for on-line and off-line part were chosen as $\mu_{\text{on}} = \mu_{\text{off}} = 0.002$ with $\lambda = 0.2$. It can be seen in Fig. 3 that the algorithm is robust against speaker movements and converges quickly due to the block-on-line structure.

Our scalable real-time system is implemented on a regular laptop using C++ in combination with the efficient Intel Integrated Performance Primitives (IPP) library. The demonstrator is applicable to $P \times P$ scenarios ($P = 2, 3, \ldots$) and works both under Linux and Windows operating systems. The computational load on an 1.6 GHz Intel Pentium 4 Processor for the above-mentioned parameter settings is approximately 70%. A video showing the capability of the system in reverberant rooms can be found at `www.LNT.de/~aichner/bss_video.html`

4 Conclusions

In this paper we presented a real-time implementation of an efficient BSS algorithm based on a general class of broadband algorithms. The system is robust to speaker movements and exhibits a low latency, showing the applicability of this method to real-world scenarios.

References

1. R. Aichner, S. Araki, S. Makino, T. Nishikawa, and H. Saruwatari. Time-domain blind source separation of non-stationary convolved signals by utilizing geometric beamforming. In *Proc. Neural Networks for Signal Processing*, pp 445–454, 2002.
2. H. Buchner, R. Aichner, and W. Kellermann. A generalization of a class of blind source separation algorithms for convolutive mixtures. In *Proc. Int. Symp. on Independent Comp. Analysis and Blind Signal Separation (ICA)*, pp 945–950, 2003.
3. H. Buchner, R. Aichner, and W. Kellermann. Blind source separation for convolutive mixtures: A unified treatment. In J. Benesty and Y. Huang, editors, *Audio Signal Processing for Next-Generation Multimedia Communication Systems*. Kluwer Academic Publishers, Boston, Feb. 2004.

4. J.S. Garofolo et al. TIMIT acoustic-phonetic continuous speech corpus, 1993.
5. S. Haykin. *Adaptive Filter Theory*. Prentice Hall Inc., Englewood Cliffs, NJ, 4th edition, 2002.
6. A. Hyvaerinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, 2001.
7. J.D. Markel and A.H. Gray. *Linear Prediction of Speech*. Springer, Berlin, 1976.
8. K. Matsuoka, M. Ohya, and M. Kawamoto. Neural net for blind separation of nonstationary signals. *IEEE Trans. Neural Networks*, 8(3):411–419, 1995.
9. R. Mukai, H. Sawada, S. Araki, and S. Makino. Robust real-time blind source separation for moving speakers using blockwise ICA and residual crosstalk subtraction. In *Proc. ICA*, pages 975–980, 2003.
10. T. Nishikawa, H. Saruwatari, and K. Shikano. Comparison of time-domain ICA, frequency-domain ICA and multistage ICA for blind source separation. In *Proc. European Signal Processing Conference*, volume 2, pages 15–18, Sep. 2002.