Realtime realization aspects of the CoBliSS blind signal separation algorithm

Realtime Realization Aspects of the CoBliSS Blind Signal Separation Algorithm

Daniël W.E. Schobben and Piet C.W. Sommen

Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
Tel. +31/402472366 Fax. +31/402455674
Email: D.W.E.Schobben@ele.tue.nl and P.C.W.Sommen@ele.tue.nl
URL: http://www.esp.ele.tue.nl/~daniels/

ABSTRACT

Recently, the Convolutive Blind Signal Separation algorithm (CoBliSS) was introduced. CoBliSS is based on second order statics only and is able to control a multichannel filter with thousands of tabs as is required in acoustical applications. In this paper the feasibility of a real-time implementation of the CoBliSS algorithm is investigated. An efficient implementation is proposed and the corresponding computational complexity is discussed.

1. INTRODUCTION

Blind Signal Separation (BSS) is the process that aims at separating a number of source signals from observed mixtures of those sources. For example, in an acoustical application, these mixtures might originate from a recording using multiple microphones. The term "blind" indicates that both the mixing and the sources are unknown. The recently introduced CoBliSS Blind Signal Separation (BSS) algorithm [1] is capable of separating real sound sources in a real acoustical environment. Current implementations of the algorithm operate however still off-line. This paper presents an overview of the computational complexity of the algorithm components. It is shown that a real-time implementation is feasible.

2. NOTATION

Throughout, time and frequency signals will be denoted by lower case and upper case characters respectively. A character which denotes a vector will be underlined. Superscripts denote the vector or matrix dimensions, a matrix with one superscript is square. Also, A^*, A^T, A^H and A^{-1} denote complex conjugate, matrix transpose, hermitian transpose and matrix inverse respectively and $\mathrm{j}^2 = -1$. Element-wise multiplication is denoted by $\otimes$. The expectation operator will be denoted by $E\{.\}$. The $N \times N$ identity matrix and the $K \times L$ zero matrix will be denoted by $\mathbf{I}^N$ and $\mathbf{0}^{K,L}$ respectively. The k, l^{th} element of matrix A and the l^{th} element of vector $\underline{B}$ is denoted as $(A)_{kl}$ and $(\underline{B})_l$ respectively. The $M \times M$ Fourier matrix $\mathcal{F}^M$ is defined as $(\mathcal{F}^M)_{kl} = e^{\frac{-2\mathrm{j}\pi kl}{M}}$. The matrix square root sqrtm(.) is defined as $A = \text{sqrtm}(B) \Leftrightarrow A^H A = B$. Moreover, $A^H = A$ when B a complex symmetric matrix, i.e. $B^H = B$. The $N \times N$ mirror matrix $\mathbf{J}^N$ has ones on its anti-diagonal and zeros elsewhere. Time indices are not mentioned explicitly in all equations.

The notation is in accordance to Fig. 1 which depicts the mixing/unmixing system. The independent sources $s_1 \ldots s_J$ are mixed by the mixing system H to obtain the microphone signals $x_1 \ldots x_J$. Throughout, both the number of sources and the number of sensors are equal to J. Time indexes are not mentioned explicitly in all formulas.

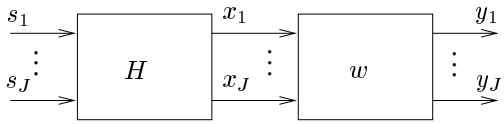

Figure 1: Cascaded mixing/unmixing system

3. THE COBLISS ALGORITHM

Both the number of sources and the number of microphones are assumed to equal J. The CoBliSS algorithm consists of the following steps [1];

1. Blocks of microphone signals are transformed to the frequency domain $\forall a$:
$$\underline{X}_a^M = \mathcal{F}^M \begin{pmatrix} x_a[nB - M + 1] \\ \vdots \\ x_a[nB] \end{pmatrix}$$
The blocks are of length M and are overlapping; every block contains only B new samples.

2. The cross-correlation estimates are updated efficiently in the frequency domain $\forall a, c$:
$\breve{\underline{R}}_{ac}^M := \alpha \breve{\underline{R}}_{ac}^M + (1-\alpha)\left((\underline{X}_a^M)^* \otimes \underline{X}_c^M\right)$
The forgetting factor α may vary from 0 to 1 depending on the application. Usually α is chosen near to 1, e.g. $\alpha = 0.99$.

3. When the cross-correlation matrices are updated several times the weights are initialized by decomposing $\breve{R}_p^J$ using the matrix square root
$\forall p : \breve{W}_p^J = \text{sqrtm}(\breve{R}_p^J)^*$
Note that the $(\breve{R}_p^J)_{a,c} = (\breve{\underline{R}}_{ac}^M)_p$.

4. The weights are constraint so that they correspond to linear Finite Impulse Response (FIR) filters of length N $\forall p$:
$\underline{\breve{W}}_{jc}^{M} := \mathcal{F}^{M} \begin{pmatrix} \mathbf{I}^{N} & \mathbf{0}^{N,M-N} \\ \mathbf{0}^{M-N,N} & \mathbf{0}^{M-N} \end{pmatrix} (\mathcal{F}^{M})^{-1} \underline{\breve{W}}_{jc}^{M}$
Note that the $(\breve{W}_p^J)_{a,c} = (\underline{\breve{W}}_{ac}^{M})_p$.

5. The filtering is performed efficiently in the frequency domain using the overlap-save method [2] to obtain the separated outputs
$\underline{y}_j^B = (\mathbf{0}^{B,M-B}\mathbf{I}^B)(\mathcal{F}^M)^{-1} \sum_{a=1}^{J} (\underline{X}_a^M \otimes \underline{\breve{W}}_{ja}^{M})$

6. The weights are modified such that the algorithm produces uncorrelated outputs (all matrices are of size $J \times J$)
$\forall p : \breve{W}_p := \breve{W}_p C_p$ with
$C_p = (\text{sqrtm}(\breve{W}_p^T \breve{W}_p^*)^*)^{-1} \text{sqrtm}(\breve{R}_p^{-1})^*$

7. The weight matrices are normalized using the l_2 norm
$\breve{W}_p^J := \frac{\breve{W}_p^J}{||\breve{W}_p^J||}$

8. All items are repeated iteratively except for the initialization in item 3.

A computational expensive part of the algorithm is the matrix square root that is required for the weight update. However, once the estimations of the cross-correlation matrices $\underline{\breve{R}}_{ac}^{M}$ converge, the weight update can be approximated by [1]

$$\breve{W}_p'^J = \breve{W}_p^J (\mathbf{I}^J - \tfrac{1}{2}\Delta\breve{R}_p^J (\breve{W}_p^J)^H \breve{W}_p^J) \qquad (1)$$

with $\Delta\breve{R}_p$ is the updated $\breve{R}_p$ minus the previous estimate of $\breve{R}_p$. This is a fast approximation of the weight update which does not require the computational intensive matrix square root.

4. COMPUTATIONAL COMPLEXITY

In this section, the computational complexity of the CoBliSS algorithm is discussed. The fast weight update will be considered to determine the overall complexity of CoBliSS. In the sequel it is assumed that a complex multiplication is performed using 4 real multiplications (muls) and 2 real additions (adds). As a measure of complexity, the number is floating point operations is used which is assumed to be equal to the sum of real muls and adds. In the Fourier domain, there are $M/2 - 1$ frequency bins with complex data and 2 frequency bins with real numbers. In the sequel $M/2$ complex bins are considered instead for the sake of simplicity. This is justified as in typical cases M is large, e.g. $M = 2048$, and J is small, e.g. $J = 2$.

4.1. Fast Fourier Transforms

In the CoBliSS algorithm, windowed Fast Fourier Transforms (FFTs) can be used. Windowed FFTs are used to transform data which is augmented with zeros more efficiently. Windowed inverse FFTs can be used when a part of the time-domain data is irrelevant. The input data of CoBliSS consist of overlapping blocks of data; each block of length M contains only B new samples. FFTs of overlapping data can also be done using windowed FFTs [2]. The computational complexity of the FFTs that transform the data to the frequency domain is $J\frac{5}{2}M\log_2(B)$ flops. The same complexity is needed for transforming the filtered data back to the time domain, as only B output samples are relevant. Also, $2J^2$ (I)FFTs are needed to constrain the weights of the filters. This requires $2J^2\frac{5}{2}M\log_2(N)$. For CoBliSS, $B = N$ is a common choice so that the (I)FFTs require a total of $(J^2 + J)5M\log_2(N)$ flops.

4.2. Correlation Update

In the calculation of the correlation update, two aspects are exploited. One is that the correlation update

$$\underline{\breve{R}}_{ac}^{M} := \alpha\underline{\breve{R}}_{ac}^{M} + (1-\alpha)\left((\underline{X}_a^M)^* \otimes \underline{X}_c^M\right)$$

is symmetrical, i.e. $\underline{\breve{R}}_{ac}^{M} = (\underline{\breve{R}}_{ca}^{M})^*$, as can be seen from it's definition. The other is that it is real for $a = c$. The complexity required to element-wise multiply the $\frac{J(J-1)}{2}\frac{M}{2}$ complex elements ($a \neq c$) is $\frac{J(J-1)}{2}\frac{M}{2}$ complex muls. The complexity required to compute the $J\frac{M}{2}$ real elements ($a = c$) is $2J\frac{M}{2}$ real muls and $J\frac{M}{2}$ real adds.

Multiplying with $(1-\alpha)$ takes $J\frac{M}{2}$ real muls for the real elements ($a = c$) and $\frac{J(J-1)}{2}2\frac{M}{2}$ real muls for the complex elements ($a \neq c$). The same complexity is required to calculate the product $\alpha\underline{\breve{R}}_{ac}^{M}$. Also, the results must be added, which takes $J\frac{M}{2} + \frac{J(J-1)}{2}2\frac{M}{2}$ real adds.

In total, $2J^2M$ real muls and J^2M real adds are required for the correlation update.

4.3. Weight Update

In the fast weight update the symmetrical matrix $(\breve{W}_p^J)^H\breve{W}_p^J$ is calculated which is multiplied with the symmetrical matrix $\Delta\breve{R}_p^J$. The multiplication with $\frac{1}{2}$ can readily be combined with $(1-\alpha)$ in the correlation update. The result is subtracted from the identity matrix and then multiplied with $\breve{W}_p^J$.

First, consider the product

$$((\breve{W}_p^J)^H\breve{W}_p^J)_{k,l} = \sum_{m=1}^{J} (\breve{W}_p^J)_{m,k}^* (\breve{W}_p^J)_{m,l}$$

For the the diagonal elements ($k = l$), $2J^2$ real muls and $2J^2 - J$ real adds are required. For the off-diagonal elements ($k \neq l$), $\frac{J(J^2-J)}{2}$ complex muls and $(J-1)\frac{J^2-J}{2}$ complex adds are required. In total, $2J^3$ real muls and $2J^3 - J^2$ real adds are needed to calculated $(\breve{W}_p^J)^H\breve{W}_p^J$.

The matrix subtraction that yields $\Delta\breve{R}_p$ takes J real adds for the diagonal elements and $J\frac{(J-1)}{2}$ complex adds for the off-diagonal elements; J^2 real adds in total. The multiplication of the symmetrical matrices $\Delta\breve{R}_p^J$ and $(\breve{W}_p^J)^H\breve{W}_p^J$ takes $4J^3 - 6J^2 + 3J$ real muls and $4J^3 - 7J^2 + 2J$ real adds (see appendix A). To subtract the result from the identity matrix takes J real adds. In total, the weight update requires $6J^3 - 6J^2 + 3J$ real muls and $6J^3 - 7J^2 + 2J$ real adds.

4.4. Filtering

The filtering is performed efficiently in the frequency domain using the overlap-save technique. Besides the inverse FFTs, the filtering requires element-wise vector products. This takes $\frac{M}{2}J^2$ complex muls and $\frac{M}{2}(J-1)J$ complex adds.

4.5. Normalization

The $\breve{W}_p$ are normalized to prevent them from collapsing due to extreme high or low energy of the input signals for frequency bin p. To calculate the l_2 norm of $\breve{W}_p$, $2J^2$ real muls are needed to calculated the magnitude of the elements of $\breve{W}_p$. Also, J^2 real adds are needed to add these magnitudes and one square root is required to calculate the norm. Next, all elements of $\breve{W}_p$ must be divided by the norm, which takes $2J^2$ real muls. In total, the normalization takes $4J^2$ real muls and J^2-1 real adds and 1 square root.

4.6. Overview

The weight update and the normalization must be done for all $M/2$ frequency bins. An overview of the total complexity that is required compute one recursion of the CoBliSS algorithm in is given in Table 1. A typical numerical example

Table 1: Computational Complexity Overview

Operation	Item	Complexity (flops)
(I)FFTs	1. 6. 7.	$(J^2+J)\cdot 5M\log_2(B)$
Correlation update	2.	$3J^2M$
Weight update	4.	$\frac{M}{2}(12J^3-13J^2+5J)$
Normalization	5.	$\frac{M}{2}5J^2$
Filtering	7.	$\frac{M}{2}(2J^2-J)$

is: $M=2048$ (transform size), $B=1024$ (number of new samples each block), $f_s=44.1$kHz (sample rate), $J=2$ (2 microphones). The computational complexity evaluated for these parameters amounts 29 Mega-flops. The complexity is also evaluated as a function of the number of microphones J for different transform sizes M as shown in Figure 2. The sample frequency $f_s=44.1$kHz for this figure. Note that the complexity of the algorithm depends linearly on the sample frequency. State-of-the-art hardware (e.g. multi-DSP platforms) can handle about 1 Gigaflops per second. A typical personal computer achieves 200 Megaflops per second. Therefore a real-time implementation of CoBliSS is feasible, even for more than 2 microphones.

Most of the complexity is due to the FFTs. Figure 3 shows both the overall complexity and the complexity due to the FFTs. Again, increasing transform sizes correspond to higher complexity curves. Up to 10 microphones ($J=10$) the FFTs take most of the total complexity. Therefore, the complexity depends only quadratically on the number of microphones. For more than 10 microphones, the CoBliSS weight update becomes computationally intensive. The complexity is than proportional with J^3 and real-time operation is no longer feasible.

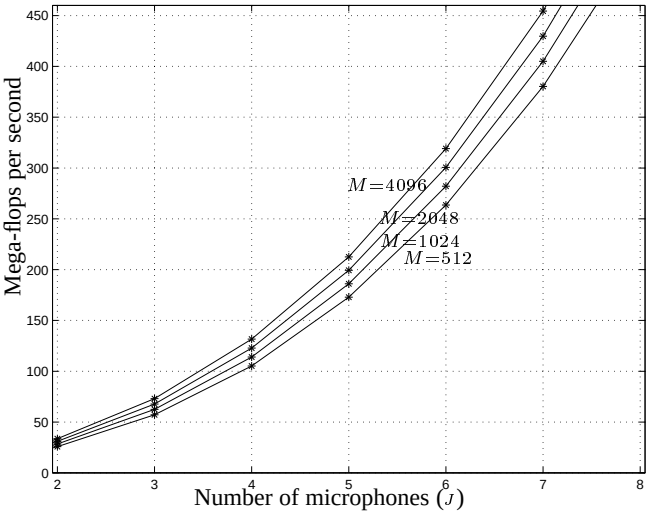

Figure 2: Overall complexity of the CoBliSS algorithm as a function of the number of microphones J and the transform size M, with $f_s=44.1$kHz

5. CONCLUSIONS

The computational complexity of the recently introduced CoBliSS algorithm is investigated in this paper. It is shown that a real-time implementation of CoBliSS is feasible at a typical personal computer, even for more than 2 microphones. The computational complexity depends mainly on the FFTs in the algorithm which is quadratically dependent on the number of microphones.

More information about the CoBliSS algorithm including audio examples can be found at

`http://www.esp.ele.tue.nl/~daniels`

Appendix A: Product of Complex Symmetrical Matrices

The product of two complex symmetrical matrices can be written as

$$c_{kl}=\sum_{m=1}^{J}a_{km}b_{ml} \text{ with } a_{km}=a_{mk}^* \text{ and } b_{ml}=b_{lm}^*$$

In the calculation of each of the J^2-J off-diagonal elements there are 2 real numbers involved (a_{kk} and b_{kk} are real $\forall k$) so that only $J-2$ complex muls and 4 real muls are required. Also, $J-1$ complex additions are needed for the summation. In the calculation of each of the J diagonal elements one real mul and $J-1$ complex muls are needed. All complex muls are encountered twice in the calculation of the diagonal elements, so that only half of these need to be computed. Also, J real adds and $(J-2)J$ complex adds are needed to sum the results. In total, $4J^3-6J^2+3J$ real muls and $4J^3-7J^2+2J$ real adds are needed per frequency bin. If the matrix symmetries would not have been exploited, a total of $4J^3$ real muls and $4J^3-2J^2$ would have been required. Hence, a significant improvement is achieved for small J.

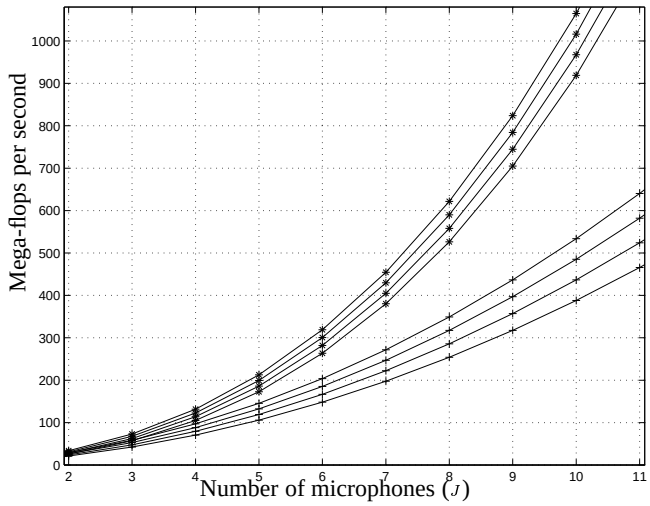

Figure 3: Overall complexity ('*') and the complexity due to the FFTs ('+') of the CoBliSS algorithm as a function of the number of microphones J and the transform size M, with $f_s = 44.1$kHz

6. REFERENCES

[1] D.W.E. Schobben and P.C.W. Sommen. A new convolutive blind signal separation algorithm based on second order statistics. In *Proc. of the Int. Conf. on Signal and Image Proc.*, Oct. 1998.

[2] D.W.E. Schobben, G.P.M. Egelmeers, and P.C.W. Sommen. Efficient realization of the block frequency domain adaptive filter. In *ICASSP 97 : Int. Conf. on Acoustics, Speech and Signal Proc.*, pages 2257–2260, Apr. 1997.