

***Adaptive Blind Audio Signal Separation System using a
TI DSP***

*J. Peters, J van de Laar, E. Habets and P. Lökkart (student members)
P. Sommen (advising professor)*

April 30, 2001

**Signal Processing Systems Group (SPS),
Department of Electrical Engineering**
Web site: <http://www.sps.ele.tue.nl>

**Technische Universiteit Eindhoven (TU/e)
The Netherlands**

Adaptive Blind Audio Signal Separation algorithm using a TI DSP

ABSTRACT

Blind Signal Separation (BSS) deals with the problem of separating independent sources from their observed mixtures while both the mixing process and original sources are unknown. Examples of BSS algorithms employed in acoustical applications can be found among others in audio teleconferencing systems, hearing aids, noise cancelling etc. Such systems usually involve speakers that are at some unknown, time varying, distances from the microphones. Therefore the recorded microphone signals contain the unknown original sources mixed by unknown acoustical transfer functions. The main objective of such a system is to produce speech with a high-quality intelligibility despite a low signal-noise ratio of the observations. Above that, in contrast to many other techniques (e.g. beamforming) the blind approach is independent of the actual source locations. In this project such a high quality is achieved by using algorithms that operate across the entire audio spectrum. The complexity of the entire system is related to the room acoustics. The acoustic model of a room is very complex. In addition, due to the continuously changing nature of the acoustics, the (un)mixing system is time varying. Therefore, adaptive filters are required.

Recently in our research group a new adaptive blind signal separation algorithm, which still is in research, has been introduced into literature [Yin]. This algorithm is based on a simplified mixing model and second order statistics. The simplified mixing model uses the property that the acoustic transfer functions from a source to two closely spaced microphones are very similar, therefore only a difference between these two functions is needed. The algorithm is efficiently realized in frequency domain.

The main purpose of this project is to make a realization of this A a d a p t i v e Bl i n d A u d i o S i g n a l S e p a r a t i o n a T e x a s I n s t r u m e n t s (T I ™) T M D 3 2 6 0 0 6 7 0 1 E v a l u a t i o n (E V M) equipped with the TMS320C6701 floating point DSP, using Code Composer Studio as software development tool. With this realization the new algorithm can be developed further because realistic experiments can be done in real time. In order to do these experiments, also a flexible and user friendly Graphical User Interface (GUI) has been built. The GUI makes use of the Real Time Data eXchange (RTDX) protocol developed by TI. Experiments show that with A-BLASS-TI two independent simultaneously occurring audio signals can be separated in real time.

“This document was an entry in the TI DSP Challenge 2000, an annual contest organized by TI to encourage students from around the world to find innovative ways to use DSPs. For more information on the TI DSP Challenge 2000, see TI’s World Wide Web site at www.ti.com/sc/dsp_challenge.”

Contents

TABLE OF FIGURES	3
TABLES	3
LIST OF ABBREVIATIONS	4
1. INTRODUCTION	5
2. ALGORITHM DESCRIPTION.....	6
2.1 SIMPLIFIED MIXING MODEL (SMM)	6
2.2 OVERLAP-SAVE IN FREQUENCY DOMAIN	7
2.3 ADAPTATION RULE	8
2.4 MAIN STEPS AND ADVANTAGES OF A-BLASS-TI.....	9
3. ALGORITHM IMPLEMENTATION	11
3.1 INTRODUCTION.....	11
3.2 SOFTWARE IMPLEMENTATION	11
4. GRAPHICAL USER INTERFACE.....	13
4.1 INTRODUCTION.....	13
4.2 COMMUNICATION.....	13
4.3 THE GRAPHICAL USER INTERFACE.....	13
5. EXPERIMENTAL RESULTS	15
6. CONCLUSIONS AND FUTURE RESEARCH.....	17
APPENDIX A : RUNNING THE DEMO.....	19
APPENDIX B : SOURCE CODE A-BLASS-TI	20
APPENDIX C : SOURCE CODE GUI.....	20

Table of figures

Figure 1: General system.....	5
Figure 2: Simplified Mixing Model (SMM).....	6
Figure 3: Demixing structure for each separate frequency bin.....	7
Figure 4: Basic overlap-save operations in frequency domain.....	8
Figure 5: Blockdiagram of A-BLASS-TI	12
Figure 6: Communication channels.....	13
Figure 7: The Graphical User Interface	14
Figure 8: Decrease of cost function as a function of time for real world mixtures.....	15
Figure 9: Decrease of cost function as function of time for a complex real world mixture.....	16

Tables

Table 1: List of wave-files used to test A-BLASS-TI	14
Table 2: Renaming wave files	19

List of Abbreviations

A-BLASS-TI	Adaptive-Blind Audio Signal Separation algorithm- using a Texas Instruments DSP
BSS	Blind Signal Separation
DRIR	Difference Room Impulse Response
DSP	Digital Signal Processor
EVM	EValuation Module
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
IFFT	Inverse Fast Fourier Transform
ISR	Interrupt Service Routine
GUI	Graphical User Interface
PCI	Peripheral Component Interconnect
RIR	Room Impulse Response
RTDX	Real Time Data eXchange
SMM	Simplified Mixing Model
TI™	Texas Instruments

1. Introduction

Blind Signal Separation (BSS) deals with the problem of separating independent sources only from their observed mixtures while both the mixing process and original sources are unknown. Examples of BSS algorithms employed in acoustic applications can be found for example in audio teleconferencing systems, hearing aids, noise cancelling etc. The main objective of such systems is, despite a low signal-to-noise ratio of the observations, to extract a desired individual speech signal with a high-quality intelligibility when several speakers are talking simultaneously or when the desired speech signal is strongly attenuated by background noise. Above that, in contrast to many other techniques (e.g. beamforming) the blind approach is independent of the actual source locations. Both the acoustic transfer functions and the source signals are unknown and the available microphone signals contain a mixture of the sources. In order to extract the sources from these observations use is made of the fact that the source signals are mutually statistically uncorrelated. The complexity of the entire system is related to the room acoustics. The acoustic model of a room is very complex. In addition, due to the continuously changing nature of the acoustics, the (un)mixing system is time varying. Therefore, adaptive filters are required. For complexity reasons we restricted ourselves in this project to the case of two sources, two microphones and two output signals as depicted in Figure 1.

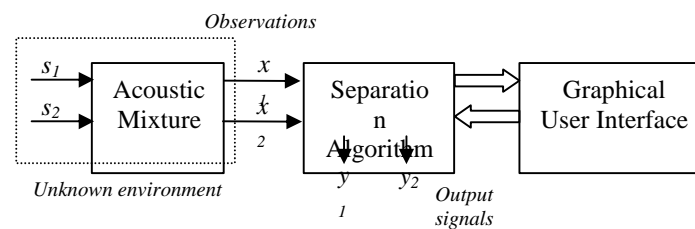


Figure 1: General system

A growing number of researchers have been attracted to the BSS problem mainly concentrating on theory and simulations. In many cases the most important step, i.e. namely the evaluation by realistic experiments, remains undone. In order to develop a new algorithm that can be used in real world situations, these evaluations are indispensable. In this project we realized a new Adaptive BLind Audio Signal Separation algorithm on a Texas Instruments TMS320C6701 DSP system (A-BLASS-TI). With this system we are able to perform realistic experiments that are necessary for the further development of this new algorithm. Among others a user-friendly graphical user interface has been built for this purpose.

In order to realize such a complex BSS system, the philosophy of A-BLASS-TI is based on efficiency and simplicity. Efficiency has been obtained by using an appropriate way of block processing in the frequency domain, which results in high quality intelligibility across the entire audio spectrum. Simplicity has been obtained on the one hand by using a simplified model. The acoustic transfer functions from a single source to two closely spaced microphones are very similar. Only a difference between these two transfer functions is needed. On the other hand simplicity has been used in the design of the adaptation rule. Due to non-stationarity of the source signals, only second order statistics are sufficient [Parra] and have been used. The gradient descent adaptation algorithm minimizes a cost function that is directly related to the cross correlation between the output signals.

2. Algorithm Description

In this chapter a global description of the algorithm is given. First the background and advantages of the simplified model are given, then the overlap-save procedure in frequency domain is discussed, while the third section deals with the construction of the update rule for the adaptation process. In the last section the main steps and advantages of A-BLASS-TI are summarized.

2.1 Simplified Mixing Model (SMM)

The mixing process that takes place in the acoustic environment from two sources to two microphones contains four different acoustic impulse responses h_{ij} , with h_{ij} the acoustic impulse response from source j to microphone i . As an example microphone signal x_i equals the sum of source s_j that is convolved with h_{i1} plus source s_2 convolved with h_{i2} . This is depicted on the left hand side of Figure 2.

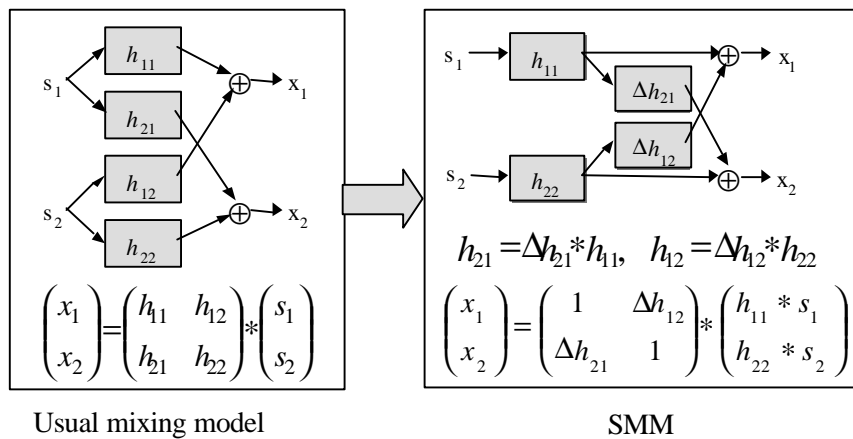


Figure 2: Simplified Mixing Model (SMM)

Demixing of this process with a BSS algorithm needs the inversion of a matrix containing these four different acoustic impulse responses h_{ij} . The result of this demixing process is an estimate of the original sources s_1 and s_2 . This demixing process can be reduced in complexity by focussing on the difference impulse responses rather than on the impulse responses themselves. This is shown in the right hand side of Figure 2. In this case the demixing process only needs to take care of a matrix containing two difference impulse responses Δh_{21} and Δh_{12} . Above that, the resulting output signals of this simplified approach are estimates of the source signals as they sound in the neighbourhood of the microphones $s_1 * h_{11}$ and $s_2 * h_{22}$ respectively. These output signals are more natural since each of them reflects what we expect to hear at the position of a microphone if only one source would have been present.

As mentioned before, the algorithm, and thus the demixing structure, is implemented in the frequency domain such that convolutions can be performed as multiplications. Since the frequency bins are more or less independent from each other, from now on we mainly focus on one frequency bin in the main part of the discussion for simplicity. As a result, the convolutive time domain mixtures are replaced by instantaneous (scalar) frequency domain mixtures. By using upper case letters for the frequency domain the simplified mixing/demixing process can be described for each frequency bin according the following equation:

$$\begin{aligned} \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} &= \begin{pmatrix} 1 & \Delta H_{12} \\ \Delta H_{21} & 1 \end{pmatrix} \bullet \begin{pmatrix} \tilde{S}_1 \\ \tilde{S}_2 \end{pmatrix} \Rightarrow \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} = \begin{pmatrix} 1 & \Delta \hat{H}_{12} \\ \Delta \hat{H}_{21} & 1 \end{pmatrix}^{-1} \bullet \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \\ &= G \cdot \begin{pmatrix} 1 & -\Delta \hat{H}_{12} \\ -\Delta \hat{H}_{21} & 1 \end{pmatrix} \bullet \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \end{aligned} \quad (1)$$

with

$$Y_i \text{ estimate of } \tilde{S}_i = S_i \bullet H_{ii} \text{ and factor } G = 1 / (1 - \Delta \hat{H}_{21} \cdot \Delta \hat{H}_{12})$$

This simplified demixing structure is equal for each separate frequency bin and is depicted in Figure 3.

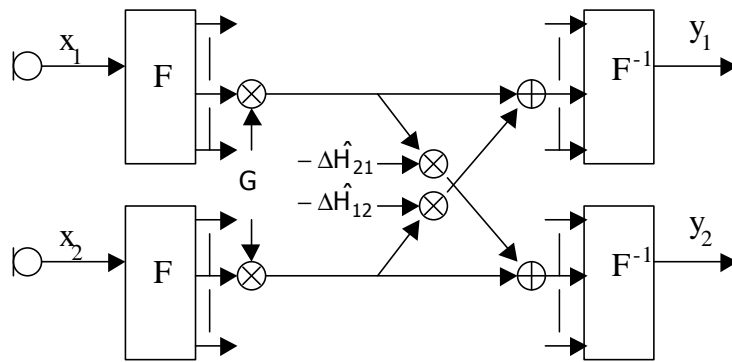


Figure 3: Demixing structure for each separate frequency bin

By using an underlined letter to denote a vector, the mixing and demixing of Eq. (1) can be written in matrix-vector notation for each separate frequency bin, as follows:

$$\underline{X} = D \bullet \underline{\tilde{S}} \Rightarrow \underline{Y} = \underline{\hat{D}}^{-1} \bullet \underline{X} \text{ is estimate of } \underline{\tilde{S}} \quad (2)$$

2.2 Overlap-save in frequency domain

From literature [Oppenheim] it is known that a filtering, or convolution operation in time domain can be performed efficiently in frequency domain by a complex multiplication. In order to meet this efficiency an appropriate way of block processing and the use of Fast Fourier Transforms (FFT) has to be applied. For adaptive filters this results in the use of overlap-save method [Sommen]. Figure 4 gives the basic steps for the overlap-save method when applied to fixed filters.

The first step to meet efficiency is to apply block processing. The ‘infinite’ long input signal x is cut into blocks. Each of these blocks is convolved successively with the impulse response w , resulting in blocks of the output signal y . The convolution, which is a linear operation, is performed in the frequency domain. The transformation is done by the use of FFT’s, which are cyclic operations. In order to perform a linear convolution with a cyclic one the input blocks need an overlap (typically 50%), the impulse response weights have to be appended with zeros and from the resulting output blocks only a part is a clean linear convolution result. When applying this technique to adaptive filters we have to realize that, although the update will take place in the frequency domain, an extra constraint is necessary to push the last part of the weight vector \underline{w} to zero in the time domain.

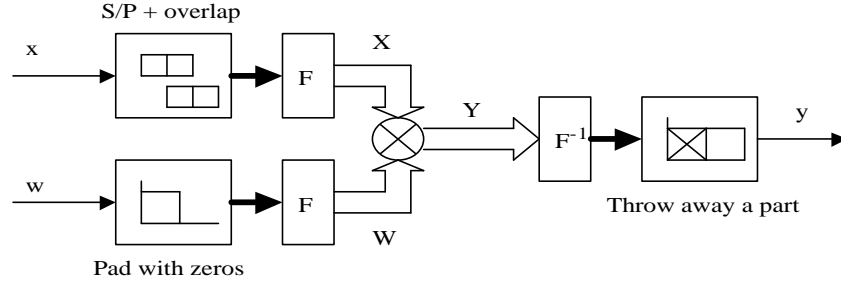


Figure 4: Basic overlap-save operations in frequency domain

As mentioned before, in the remaining part of the discussion we will mainly focus on one frequency bin. In case it is explicitly required we will denote this bin with index l . With N the length of the FFT this index l can vary between the values 0 and $N-1$.

2.3 Adaptation rule

In this section we shortly derive the update rule that is needed for the adaptation of the coefficients $\Delta\hat{H}_{ij}$. For this we make use of the following gradient descent update rule that will be applied for each frequency bin l separately (*Note: this index l is left out for simplicity reasons*):

$$\Delta\hat{H}_{ij} := \mathbf{g} \cdot \Delta\hat{H}_{ij} - \mathbf{m}_{ij} \cdot \left(\frac{\partial J}{\partial \Delta\hat{H}_{ij}} \right)^* \quad \text{for } i, j = 1, 2 \text{ and } i \neq j \quad (3)$$

in which J is a cost function that is minimized each step, \mathbf{g} is a leakage factor (typically 0.99) and \mathbf{m}_{ij} are the adaptation constants.

By using the fact that the original sources s_i are mutually statistically uncorrelated, we are able to reconstruct the original sources at the output by relating the cost function to the cross-correlation between the output sequences. With the definition of equations (1) and (2), for each separate frequency bin l , a cost function J is defined as the squared cross-power function according to:

$$J = \sum_{i \neq j} \left| \left\langle \left(\underline{Y} \cdot \underline{Y}^h \right)_{ij} \right\rangle \right|^2 = \sum_{i \neq j} \left| \left\langle \left(\hat{D}^{-1} \cdot \underline{X} \cdot \underline{X}^h \cdot \hat{D}^{-h} \right)_{ij} \right\rangle \right|^2 = \sum_{i \neq j} \left| \left(\hat{D}^{-1} \cdot R_X \cdot \hat{D}^{-h} \right)_{ij} \right|^2 \quad (4)$$

The summation is running over the non-diagonal elements $i \neq j$ of the 2×2 (squared) cross output power matrix. Furthermore the symbol h is used for complex conjugate and transpose while the symbol $\langle \cdot \rangle$ denotes an ensemble averaging operation. The 2×2 matrix R_X is the cross power spectra matrix of the input (microphone) signals.

In [Yin] it is shown that the derivative of the cost function with respect to $\Delta\hat{H}_{ij}$ is as follows:

$$\left(\frac{\partial J}{\partial \Delta\hat{H}_{ij}} \right)^* = - \sum_{p \neq q} \left\{ \left(\hat{D}^{-1} \cdot E_{ij} \cdot R_Y \right)_{qp}^* \cdot (R_Y)_{pq}^* + (R_Y)_{pq} \cdot \left(\hat{D}^{-1} \cdot E_{ij} \cdot R_Y \right)_{pq}^* \right\} \quad (5)$$

with $R_Y = \hat{D}^{-1} \cdot R_X \cdot \hat{D}^{-h}$

In this equation the 2×2 selection matrix E_{ij} is a matrix whose i,j -th element is one and the others are zero. The 2×2 cross power matrix of the input signal is estimated by means of an exponential averaging as follows:

$$R_X := \mathbf{a} \cdot R_X + (1 - \mathbf{a}) \cdot \underline{X} \cdot \underline{X}^h \quad (6)$$

where \mathbf{a} is a forgetting factor which is usually chosen close to 1 (e.g. 0.99).

From the definition of the cost function J it can be seen it is a non-linear function with respect to the adaptive weights. In theory second order statistics are not enough to find a good solution for stationary sources. In practical non-stationary situations however, it is argued in [Parra] that second order statistics can solve this problem.

Furthermore, the gradient can become very large leading to divergence of the adaptive weights. By limiting the gradient to a gate g we can overcome this problem by choosing the adaptation constant according to the following rule:

$$\mathbf{m}_{ij} = \begin{cases} g / \left| \frac{\partial J}{\partial \Delta \hat{H}_{ij}} \right| & \text{if } \left| \frac{\partial J}{\partial \Delta \hat{H}_{ij}} \right| > g \\ \mathbf{1} & \text{otherwise} \end{cases} \quad (7)$$

Also we need to take care of the fact that the weights have to perform a linear convolution. For this reason the weight vectors have to be constrained in such a way that half of the weights, in time domain, equal zero. By putting the gradients for all frequencies in a vector we can perform this constraint as follows:

$$\frac{\partial J}{\partial \Delta \hat{H}_{ij}} \equiv F \cdot Z \cdot F^{-1} \cdot \frac{\partial J}{\partial \Delta \hat{H}_{ij}} \quad (8)$$

with F the Fourier matrix and Z a diagonal matrix with $(Z)_{ii} = 0$ for that part of the weight vector that has to be constrained to zero, the other diagonal elements are 1.

Finally we note here that the permutation problem (different frequency bins of different estimates of the sources can be hustled) is solved by pushing this extra constraint on the adaptive weights each iteration [Parra].

2.4 Main steps and advantages of A-BLASS-TI

The following box summarizes the main steps of A-BLASS-TI:

- Source signals are transformed into frequency domain so that for each frequency bin the mixtures are instantaneous
- The filter coefficients are then estimated by minimizing a cost function, which is directly related to the cross-correlations between the output signals, with a gradient-based algorithm:
 - ♦ gradients are estimated as a function of the filter coefficients
 - ♦ stepsizes are calculated
 - ♦ coefficient updates are calculated
 - ♦ coefficient updates are constrained
 - ♦ filter coefficients are updated
- Filtering is performed
- Outputs are transformed back to time domain

The A-BLASS-TI philosophy of simplicity and efficiency results in the following advantages:

- Less parameters have to be estimated (length of filters can be relatively small, less filters)
- The demixed signals sound more natural because not the original sources are recovered, but the sources modified by the direct paths only
- The Algorithm can be implemented in real time
- In contrast to for instance, beamforming techniques, the blind approach is independent of the actual source locations

3. Algorithm implementation

3.1 Introduction

The algorithm outlined in the previous chapter has been implemented on a TMS320C6701 Evaluation Module board [TI]. This EVM has been installed in a PCI slot of a host PC running Windows NT 4.0 and contains all the required hardware. The CS4231A (see [Crystal] and [Carlson]) 16-bit stereo audio codec on the EVM is used to sample the two input signals and to convert the two separated output signals from digital to analogue. A-BLASS-TI uses a sample frequency of 8 kHz. The codec is equipped with a serial interface which is connected to the 'C6701 via one of the multi-channel buffered serial ports of the EVM (McBSP0).

A graphical user interface (GUI) that runs on the host PC is used to set the parameters of the program and to choose the sound signal the user wants to listen to (see next chapter). The software for implementing A-BLASS-TI on the DSP has been programmed mainly in C and is discussed in the following section. A hard copy of the source code is available as a separate document called "A_BLASS_TI_source_code.doc". In addition, it is available as a zip-file called "A_BLASS_TI_source_code.zip".

In the next section, first an overview of the main software modules that have been programmed or included is given. Finally, it is explained how the main routines work together in order to implement the A-BLASS-TI algorithm described in the previous chapter.

3.2 Software implementation

The most important functions are packed in one software module called "main_bss.c". These functions are called 'main', 'BSS' and 'Bsp0Rxlrq' and are explained in the following paragraph. A-BLASS-TI requires many complex number and vector / matrix operations. All these computations have been programmed in separate modules. The complex number operations are defined in the module "complex_calc.c" and the vector / matrix operations are in the module called "matrix_calc.c". The matrix inversion routine has been programmed very efficiently taking into account the fact that the BSS algorithm only requires the inverse a matrix that has ones on its diagonal (Eq. (1)). Some operations that are specific to 'BSS', namely the power matrix update (Eq. (6)) and the stepsize computation (Eq. (7)), are defined in the module "bss_calc.c". All hardware related initialization functions are packed into a module named "init_devices.c". In addition to the self-developed software, (assembly) modules from TI containing FFT and RTDX (Real Time Data Exchange, see next chapter) related functions are used. All software modules have associated header files with corresponding names containing relevant variable definitions and function declarations.

In the 'main' function, first the required hardware related initialization is performed (setting up the interrupt vector table, initializing the EVM board, codec, McBSP, etc). 'main' works together with the interrupt service routine (ISR) 'Bsp0Rxlrq' in order to call 'BSS' at the right moment, i.e. when a new block of input samples is available for processing. The length of a block is called N and the overlap is called B . Since a 50 % overlap is used for the overlap-save block processing, $B = N / 2$ (typically $N = 1024$). After the initialization, 'main' enters a loop that waits until B new input samples have been acquired and then the function 'BSS' is called. Figure 5 shows the flow chart of A-BLASS-TI ('BSS'). It processes one block of length N at a time and is called from 'main' with two arguments. The first argument is a pointer to a buffer of length B that contains the samples that were also used in the previous block (the 50 % overlap, see Section 2.2) and the second argument is a pointer to the buffer containing the B new input samples.

First the signals are transformed to the frequency domain by means of FFT routines from TI. A-BLASS-TI uses the first 5 blocks (this number can be modified) to make an estimation of the power matrix of the input signals x_1 and x_2 for each frequency bin by averaging over the blocks.

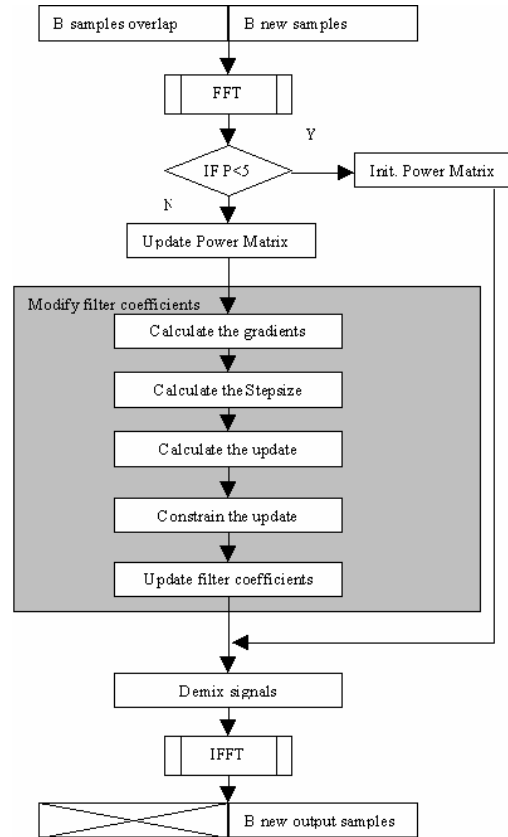


Figure 5: Blockdiagram of A-BLASS-TI

The variable P in the flowchart indicates the number of the block being processed. For each of the next blocks ($P \geq 5$), A-BLASS-TI updates the power matrix of the input signals by means of exponential averaging (Eq. (6)). This power matrix is then used for the computation of the gradients. Next, the stepsizes m_2 and m_1 for the steepest descent update rule (Eq.(7)) and the coefficient updates for the Difference Impulse Response (DIR) filters H_{12} and H_{21} are computed. Next the coefficient updates are constrained (which results in constraints on the gradients) and then the DRIR filter coefficients are updated (Eq. (3)). Finally, the BSS function demixes the input signals in the frequency domain, transforms the output signals back to the time domain and stores the last half (due to the overlap-save) of the output signals in an output buffer from where the ISR 'BspORxlrq' will send them to the line-out of the EVM board.

It should be noted that because of the fact that the input signals are real, all signals transformed to the frequency domain by means of the FFT are conjugate-symmetric and therefore all loops in the frequency domain can be halved by exploiting this property (see source code in "main_bss.c"). More precisely, only $N/2+1$ instead of N values are used (with the TI FFT routines, the first and the $N/2+1$ th transformed values are unique, and the last $N/2-1$ values can be deduced directly from the second through the $N/2$ th values).

4. Graphical User Interface

4.1 Introduction

As described in the previous chapters, the algorithm is still under development. Because of this there is a need to control and test the performance of the algorithm. This can be done with the user friendly Graphical User Interface (GUI), described in this chapter.

To set up the communication between the GUI and the DSP system, the Real Time Data eXchange protocol, developed by Texas Instruments, is used. The GUI itself has been developed with Microsoft Visual C++.

4.2 Communication

For communication with the DSP Target application, the interface sends and receives commands through two separate RTDX channels (see Figure 6). There is one channel for host-to-target communication and one for target-to-host communication. The start of a communication session is always initiated by the host application by sending a command to the target. The target will continuously poll for incoming commands and responds with an acknowledgement to the host.

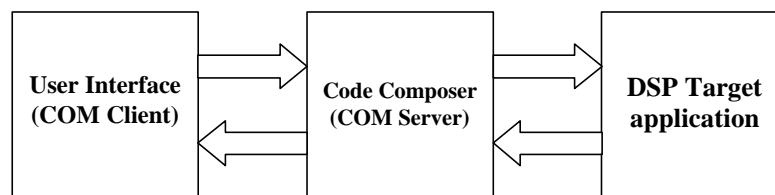


Figure 6: Communication channels

4.3 The Graphical User Interface

In the GUI, as shown in Figure 7, three main elements are distinguished. In the upper-left corner one can find a general block diagram of the algorithm. It is possible to select the output of the DSP target system, i.e. the Mixture signal or one of the output signals of the algorithm (Channel 1 or Channel 2).

The bottom part of the interface displays some status information. The control sections of the interface are explained below:

Program Control

- Start** This button will start the algorithm
- Pause** This button will halt the processing
- Stop** This button will stop and re-initialise the algorithm

Controls

- Reset** This button will set the algorithm to its initial state
- Adaptation** With this checkbox one can turn the adaptation on or off

Sound Control

One can select a predefined sound and play or stop it by using the buttons. These predefined sounds are real world mixtures, which are being used by a number of researchers working in the field of blind signal separation. Except of the sound named 'Peters', which is a self recorded audio file. All these files are available on the internet and often used as benchmarks. The table below shows the references and the filenames of the sounds.

Reference	Filenames (mono mixtures)	Internet address
Chan	P2X1.wav, P2X2.wav	http://www-sigproc.eng.cam.ac.uk/oldusers/dcbc1/research/separate.html
Lee rss	rss_mA.wav, rss_mB.wav	http://www.cnl.salk.edu/~tewon
Lee rssd	rssd_A.wav, rssd_B.wav	http://www.cnl.salk.edu/~tewon
Peters	Ex2_sm_rwm_a.wav Ex2_sm_rwm_b.wav	not published
Schobben	x1_2x2.wav, x2_2x2.wav	http://www.esp.ele.tue.nl/onderzoek/daniels/BSS.html

Table 1: List of wave-files used to test A-BLASS-TI

The original mono audio files from internet were changed into stereo audio files for use with the DSP system.

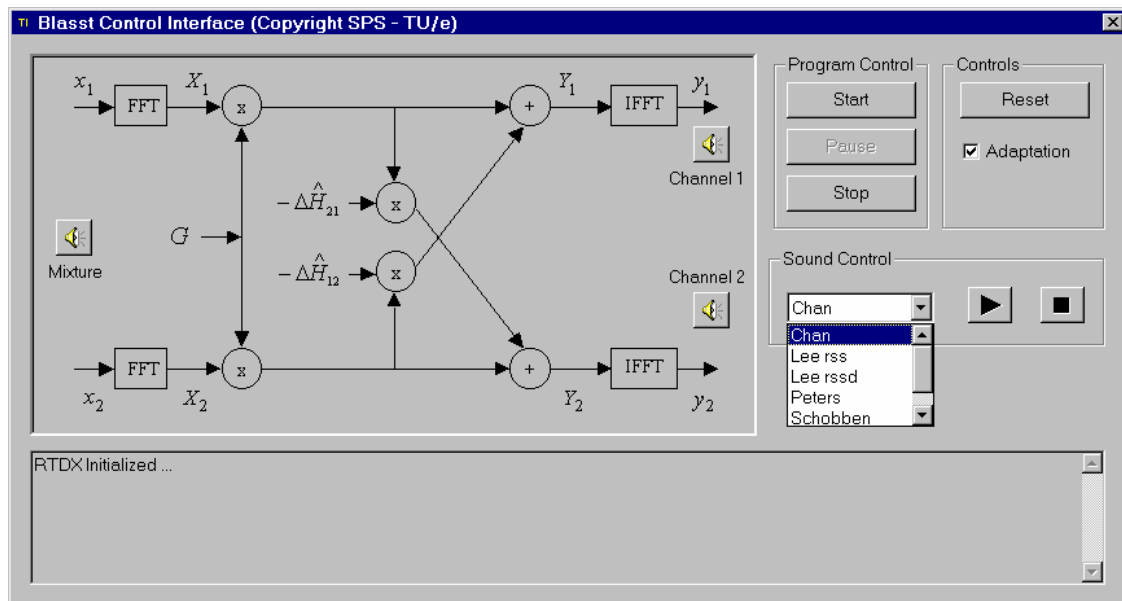


Figure 7: The Graphical User Interface

Before using this GUI, Code Composer Studio must run at the background with the target application running on the DSP system. Afterwards the interface can be used as explained above. When closing the interface, the target application will be initiated by the host to stop. The necessary steps for running a demo can be found in Appendix A.

5. Experimental Results

Experiments are done in order to evaluate our real time adaptive blind source separation system (A-BLASS-TI). Among the input signals used to test the performance of A-BLASS-TI are benchmarks of real world mixtures, recorded by Te-Won Lee, which are being used by a number of researchers working in the field of blind signal separation. Two speakers have been recorded speaking simultaneously. Speaker 1 says the digits from one to ten in English and speaker 2 counts at the same time the digits in Spanish. The recordings are made in a normal office room using a sample frequency of 16 kHz. The distance between the speakers and the microphones is about 60cm in a square ordering.

Benchmarks from Schobben and Chan have also been used to evaluate the algorithm. The conditions under which these recordings are made can be found on the corresponding internet sites (see Table 1). With these benchmarks used as input mixtures, A-BLASS-TI performs very well. Informal listening tests shows that a very good separation of the mixtures is achieved.

It must be taken into account that when listening to a recorded mixture it is more difficult to separate a desired source from background noise, than in a real world situation. A human in a noisy environment is better able to focus on a specific source, e.g. the direction of arrival of the desired source is estimated by the difference of perception for both ears. Moreover, a human perceives body language of the specific person they wants to listen to such as a mouth articulating. In a real world case all these factors helps to improve the intelligibility of a desired source.

Besides listening to the unmixed outputs, it is desirable to define the performance of A-BLASS-TI in a formal way. However this is not straightforward because it is a ‘blind’ system en therefore there is no reference signal available. The cost function J defined in (Eq. (4)) is used as a performance measure. This cost function is related to the cross-correlation between the outputs and therefore a measure for the performance of the blind signal separation algorithm.

In figure 8 the cost function ($10 \cdot \log J$) is plotted as a function of time. This is done for three benchmarks. All the results are computed with A-BLASS-TI using a FIR filter of 512 coefficients.

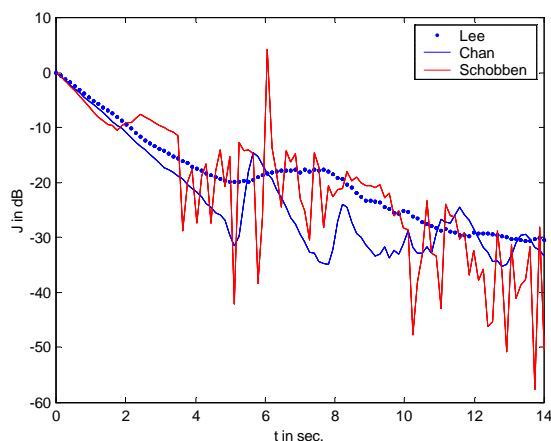


Figure 8: Decrease of cost function as a function of time for real world mixtures

It can be seen that the algorithm decreases the cost function with approximately 40 dB. The peaks occurring in the figures are due to the non-stationarity of the input signals. The initial value of the cost function is normalised to 0 dB. The reason for doing this is that the initial value of the cost function depends on the signal power of the input wav-files and the complexity of the mixtures. By doing this a straightforward comparison can be made between the different input mixtures used. It can be seen that the cost function is 20 dB decreased after approximately 3 seconds, which is an impressive reduction of the interference level. The conclusion can be drawn that the algorithm is able to track the changes in the acoustical transfer functions from the sources to the microphones.

The perceptual separation is also a very important measure for separation. Also from listening tests we can conclude that the intelligibility of one desired source is improved, due to the fact that the unwanted source is suppressed to a great extent. This holds for various kinds of real world mixtures found on the internet.

In order to test our algorithm in different environments an experimental setup is made in a room with dimensions of 4.8m by 4.8m by 2.8m (length by width by height). The acoustical behavior of the room can be changed by the use of panels on the walls with different reflection properties and the use of curtains for the windows. Two omnidirectional microphones are used which are placed at a distance of $d=10\text{cm}$ to each other for the experiment described here. The distance between the loudspeakers and the microphones is chosen to be 60cm. The distance between the loudspeakers is 50cm. In this setup a speaker and a music source are recorded simultaneously with a sample frequency of 44.1 kHz, in order to prevent aliasing. In Matlab this recording is down sampled to 8 kHz using an anti-aliasing filter. The input wav-file used for this simulation is submitted as file: Ex2_sm_st_rwm.wav. The decrease in cost function for these input files is plotted in Figure 9.

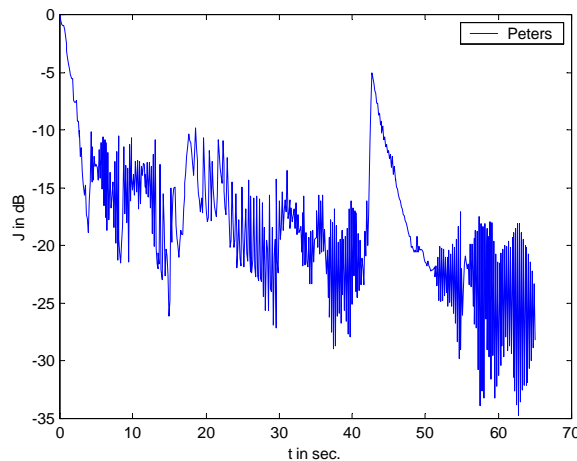


Figure 9: Decrease of cost function as function of time for a complex real world mixture

Listening tests reveal that it is still difficult to enhance a speech signal out of a mixture of music and speech, although Figure 9 shows a 25 dB decrease in the cost function. This is probably caused due to the fact that the experimental setup produces rather complex acoustical mixtures. The algorithm also assumes that the input signals are temporary stationary for a certain time. Speech signals can be seen as temporary stationary for approximately 10 ms, however for music signals that time is much shorter, which can be an explanation for the fact that the results with a speech-music mixture are not as satisfying as with the speech-speech mixtures.

6. Conclusions and future research

In this project an Adaptive BLind Audio Signal Separation algorithm has been realized on a Texas Instruments TMS320C6701 Digital Signal Processor (A-BLASS-TI). With a user friendly Graphical User Interface (GUI) this system can be used to further investigate and improve a new Blind Signal Separation (BSS) algorithm in research. With this BSS algorithm one can separate two independent simultaneously occurring audio signals. Applications can be found among others in audio teleconferencing systems, hearing aids, noise cancelling etc.

Informal listening tests of first experiments with common available benchmarks of audio mixtures, used by most researchers in the BSS field, result in very good separation. Besides these listening tests also a more objective measure has been used to evaluate and compare the different experiments. These findings are comparable with the informal listening tests.

Besides that also self generated real mixtures have been tested. The result of a mixture containing speech and noise where reasonable, but the result of speech in music was not as good compared to the benchmarks results. The main reason for this is probably that these real mixtures are much more complex as the mixtures available from the benchmarks. In order to solve these problems further research of the BSS algorithm is necessary and the result of this project, A-BLASS-TI with the GUI, can be used for this purpose very well.

Finally some points for future research using A-BLASS-TI are:

- Initialisation: use as much as possible a priori knowledge to initialize the system.
- FFT length N : evaluate the influence of N on the behavior of the system.
- Adaptation stepsize: is it possible to normalize the gradient in such a way that the adaptive behavior is independent of the statistics of the input signal?
- Tracking: How does the algorithm behave if the sources are moving.
- One source out of many: The ultimate goal of a BSS system is to separate one desired source out of many others in a real environment.

References

[Carlson]

B.G. Carlson, "TMS320C6000 McBSP Interface to the CS4231A Multimedia Audio Codec", Texas Instruments Incorporated, SPRA477, Dec. 1998

[Crystal]

Crystal Semiconductor Corporation, "CS4231 Parallel Interface, Multimedia Audio Codec", DS139PP2, Sept. 1994.

[Oppenheim]

A.V. Oppenheim and W. Schaffer, "Digital Signal Processing", New Jersey, Prentice-Hall, 1975.

[Parra]

L. Parra and C. Spence, "Convolutional Blind Separation of Non-Stationary Sources", IEEE Transactions on speech and audio processing, 2000.
Vol. 8, No. 3, p. 320-327.

[Peiyu]

Peiyu He, P.C.W. Sommen and Bin Yin, "A Real-time DSP Blind Signal Separation Experimental System Based on a New Simplified Mixing Model", Eindhoven University of Technology, submitted to Eurocon 2001, Bratislava, Slovakia, 5-7 July 2001.

[Sommen]

P.C.M. Sommen, "Adaptive filtering methodes, *On methods to use a priori information in order to reduce complexity while maintaining convergence properties*", Eindhoven University of Technology, 1992.
Doctoral dissertation.

[TI]

Texas Instruments, "TMS320C6201/6701 Evaluation Module Technical Reference", Texas Instruments Incorporated, SPRU305, Dec. 1998.

[Yin]

Bin Yin and P.C.W. Sommen, "A new convolutional blind signal separation algorithm based on second order statistics using a simplified mixing model", Eindhoven University of Technology, Proc. Eusipco '2000, European Signal Processing Conference, Tampere, Finland, 5-8 September 2000, ed. M. Gabbouj and P. Kuosmanen, 2000.

Appendix A : Running the demo

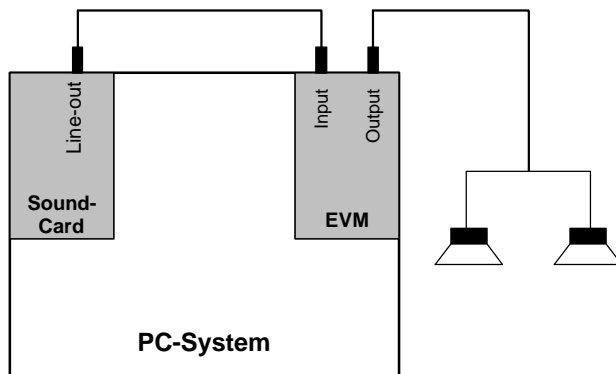
This appendix is meant as a guideline for running the BSS demo using different input wave files. The original mono audio files from the internet containing the mixtures were combined into stereo audio files for use with the DSP system. The new stereo names for these files are listed in the table below:

Reference	Filenames (mono mixtures)	Filename (stereo mixtures)
Chan	P2X1.wav, P2X2.wav	p2_ss_st_rwm.wav
Lee rss	rss_mA.wav, rss_mB.wav	rss_ss_st_rwm.wav
Lee rssid	rssd_A.wav, rssd_B.wav	rssd_ss_st_rwm.wav
Peters	Ex2_sm_rwm_a.wav Ex2_sm_rwm_b.wav	Ex2_sm_st_rwm.wav
Schobben	x1_2x2.wav, x2_2x2.wav	x_2x2_ss_st_rwm.wav

Table 2: Renaming wave files

Before running the demo, the following connections must be made (see block diagram below):

1. Make a connection between the line-out of the Sound-Card in your PC and the line-in of the EVM-board, using a 3.5 mm stereo jack-plug on both ends.
2. Connect two speakers to the output of the EVM-board.



To start the demo use the following steps:

1. <drive>:\Demo
2. Start Code Composer Studio
3. Load "main_bss.out" into the DSP.
4. Run the program.
5. Start the application "Blasst.exe".
6. Select a sound file (the file will be played automatically) .
7. Use the START button in the Program Control section to start the algorithm.

Using the speaker icons shown in the model (upper left corner), one can select an output.

Appendix B : Source code A-BLASS-TI

Appendix C : Source code GUI