REAL-TIME IMPLEMENTATION OF A SOURCE SEPARATION ALGORITHM

Matthias Baeck and Udo Zölzer

Department of Signal Processing and Communications
University of the Federal Armed Forces Hamburg, Germany
Udo.Zoelzer@unibw-hamburg.de

ABSTRACT

Source separation out of a mix of signals has been under development for many years with different approaches. We use time-frequency representations of two microphone signals to estimate the mixing parameters of the source signals. In order to evaluate the robustness of the algorithm under real-world conditions we built a real-time implementation, which is suitable to detect the sources, their mixing parameters and performs the source separation based on the mixing parameters. Our implementation only needs a few parameters and then works as a stand-alone solution with the opportunity to apply further post-processing or digital audio effects to the source signals.

1. INTRODUCTION

Source separation is an attractive tool for several audio applications. It can be used as a pre-processing technique for hearing aids [1, 2] or as a post-processing technique for demixing of stereo recordings. The main aspects are the localization of a signal of interest and the adaptation of the spatial response of an array of sensors and in our task of two sensors to achieve steering in a given direction. An overview of several source separation approaches can be found in [3, 4, 5, 6, 7]. An early approach for a two-microphone technique is based on a time-frequency representation of the microphone signals in order to perform a time-delay estimation [8]. Extensions of this basic idea are introduced in [5, 6, 9, 10, 11, 12].

For a two-channel microphone arrangement with spatially distributed sources, as shown in Fig. 1, the incoming microphone signals $x_1(n)$ and $x_2(n)$ can be written as

$$x_1(n) = \sum_{j=1}^{N} s_j(n) \tag{1}$$

$$x_2(n) = \sum_{j=1}^{N} a_j s_j(n - d_j), \tag{2}$$

where $s_j(n)$ are *N* signal sources, a_j is the relative amplitude of source *j* at microphone 2 referring to microphone 1 and d_j is the delay of source *j* between the two microphones. It should be noted that the microphone signals consist of only weighted source signals without any convolution with a room impulse response.

The main task is to detect the number N of source signals and to calculate the mixing parameters a_j and d_j corresponding to the source j in order to separate all source signals $s_j(n)$ from the two-channel signals $x_1(n)$ and $x_2(n)$. If we consider two-channel stereo recordings based on amplitude panning of several source signals without any delay adjustment, because an audio mixing console does not support a combination of amplitude and delay panning, the second parameter d_j is missing and makes it even more difficult to the separate the source signals. Nevertheless, if we have a scenario which is depicted in Fig. 1 and which can be written as in Eq. (1) and (2), the estimation of the mixing parameters is possible and a source separation can be performed.

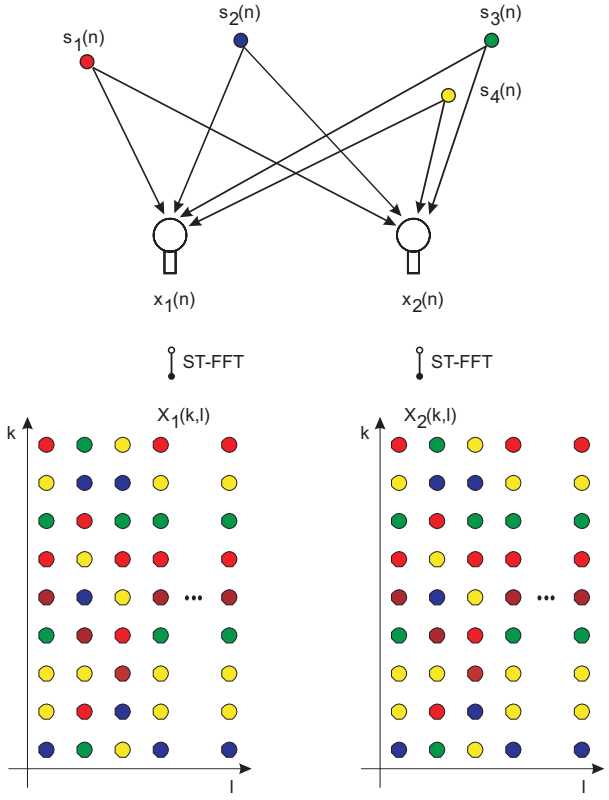

Figure 1: Microphone setup with spatially distributed sources and the time-frequency representations $X_1(k,l)$ of microphone signal $x_1(n)$ and $X_2(k,l)$ of microphone signal $x_1(n)$. The spectrogram with the two-dimensional time-frequency grid shows the basic assumption that each point on this grid belongs to only one source.

This paper will present a real-time implementation of a source separation algorithm by frequency domain processing. In section 2 we will discuss the main idea, the algorithm and some extensions. The software implementation is presented in section 3 and a performance analysis will be shown in section 4.

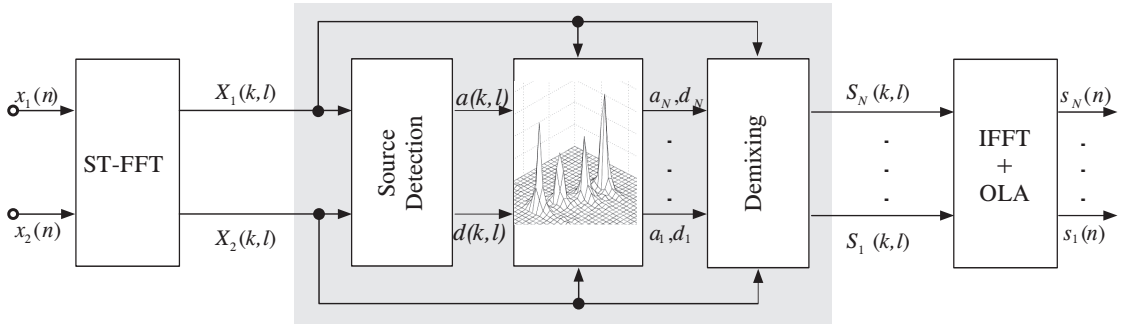

Figure 2: Block diagram of a source separation algorithm.

2. THE ALGORITHM

The DUET algorithm presented in [10, 11, 12] is based on time-frequency processing and can be implemented with a phase vocoder algorithm [13, 14, 15]. In this section we will describe the basic ideas of the source detection and source separation algorithm and will introduce some modifications and extensions [16].

2.1. Source Detection

Figure 2 shows the block diagram of the implementation which performs a source detection in a first step and a source separation in a second step. After weighting $x_1(n)$ and $x_2(n)$ with a suitable window function and performing a Fast Fourier Transform we get

$$X_1(k,l) = \sum_{j=1}^{N} S_j(k,l) \tag{3}$$

$$X_2(k,l) = \sum_{j=1}^{N} a_j S_j(k,l) e^{-j\frac{2\pi k d_j}{L}}, \tag{4}$$

where *k* is the frequency index, *l* is the time index and *L* is the length of the FFT. The most important condition is the W-disjoint orthogonality given in [10, 11, 12], which means that every point in the spectrogram corresponds to just one source (see Fig. 1). Based on this condition Equ. (3) and (4) can be simplified to

$$\begin{bmatrix} X_1(k,l) \\ X_2(k,l) \end{bmatrix} = \begin{bmatrix} 1 \\ a_j e^{-j\frac{2\pi k d_j}{L}} \end{bmatrix} S_j(k,l) \tag{5}$$

for every point (k,l). With a simple division we can now calculate a_j and d_j for every point in the time-frequency domain according to

$$(a(k,l), d(k,l)) = \left(\left| \frac{X_2(k,l)}{X_1(k,l)} \right|, \frac{L}{2\pi k} \angle \frac{X_1(k,l)}{X_2(k,l)} \right). \tag{6}$$

After calculating the mixing parameters for every point in the time-frequency domain, we can construct a histogram h which shows the distribution of the mixing parameters. A source located in the middle of the two microphones has a relative amplitude $a_j = 1$ and a delay $d_j = 0$. In order to have this central source located in the origin of the histogram, we define

$$\hat{a}(k,l) = a(k,l) - 1/a(k,l). \tag{7}$$

For a discrete histogram the parameters $\hat{a}(k,l)$ and $d(k,l)$ have to be quantized by rounding towards the nearest quantization step, which leads to the discrete values $a_q(k,l)$ and $d_q(k,l)$. Then the weighted histogram $h(\alpha,\delta)$ can be computed by adding the signal power for every point (k,l) to that point (α,δ) which corresponds to the calculated point $(a_q(k,l), d_q(k,l))$. Every source should lead to a peak in the histogram. Some post-processing on the histogram has been shown to be useful. First, the histogram should be smoothed in order to get just one single peak for each source. This supports the calculation of the correct mixing parameters. Understanding the histogram as grey scale picture, it is obvious that we can use typical methods of image processing. A binomial mask, which is well known in the area of image processing, is a good choice. Figure 3 shows the original histogram and the smoothed histogram.

The next step in a real-time processing must be an automatic peak detection [16]. We use the usual way of calculating the gradients to determine all local maxima according to the following conditions

$$\frac{\partial h}{\partial \alpha} = 0 \qquad \frac{\partial h}{\partial \delta} = 0 \tag{8}$$

$$\begin{vmatrix} \frac{\partial^2 h}{\partial \alpha^2} & \frac{\partial^2 h}{\partial \alpha \partial \delta} \\ \frac{\partial^2 h}{\partial \delta \partial \alpha} & \frac{\partial^2 h}{\partial \delta^2} \end{vmatrix} > 0. \tag{9}$$

If all conditions are valid an extreme value occurs. If $\frac{\partial^2 h}{\partial \alpha^2} < 0$ a maxima is found. The derivatives are approximated by differences to yield an efficient implementation. This peak detection delivers an approximation α_j, δ_j of the true mixing parameters a_j, d_j for each source.

2.2. Source Separation

When the mixing parameters α_j, δ_j are known we can use them to separate the sources in the demixing block shown in Fig. 2. The basic idea is to assign each point in the time-frequency domain (k,l) to one source j of the signal mix. Therefor we have to check each point of the time-frequency domain which source j belongs to that point. Under the basic assumption that for each point we have

$$X_1(k,l) = S_j(k,l) + N_1(k,l) \tag{10}$$

$$X_2(k,l) = a_j S_j(k,l) e^{-j\frac{2\pi k d_j}{L}} + N_2(k,l), \tag{11}$$

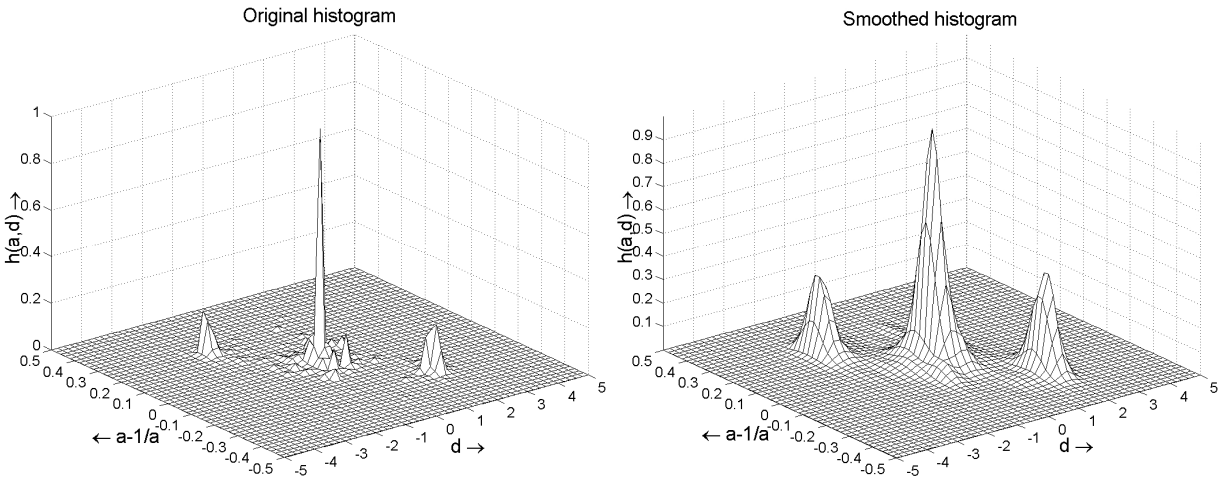

Figure 3: Original and smoothed histogram $h(a, d)$ of the mixing parameters.

where $N_1(k, l)$ and $N_2(k, l)$ are noise sources which reflect the contributions of other sources to that point, a maximum likelihood estimate can be derived to assign one source to each time-frequency point [12], based on the already estimated mixing parameters α_j, δ_j. For each source j and each point (k, l) we can calculate the likelihood function

$$L_j(k,l) = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2\sigma^2}|\alpha_j e^{-j\frac{2\pi k\delta_j}{L}} X_1(k,l) - X_2(k,l)|^2/(1+\alpha_j^2)}, \tag{12}$$

which is derived in [12]. Using the maximum likelihood we can construct a binary mask $M_j(k, l)$ to assign each point in the time-frequency domain to one source. After the IFFT and the overlap-add procedure we have all sources $s_1(n), \ldots, s_N(n)$ separated and can apply any digital audio effects to them. Re-mixing results in a mix of sources with different effects applied to different sources.

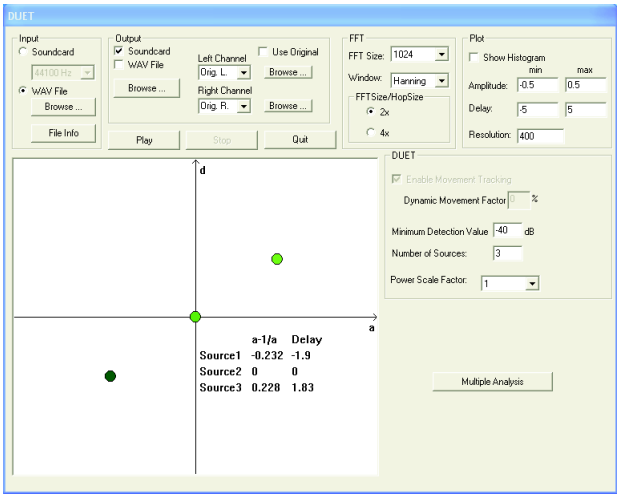

Figure 4: Graphical user interface.

3. SOFTWARE IMPLEMENTATION

The entire algorithm shown in Fig. 2 is performed by a C++ program which runs in real-time on a PC platform. The graphical user interface is depicted in Fig. 4. It allows to process a stereo signal coming from the AD converter of a sound card or a WAV file. The extracted source signals from the stereo input can be written into a WAV file or can be transferred to the DA converter of the sound card. The program allows to choose the FFT length, the hop size, and the window function. Further settings are the maximum/minimum amplitude and delay and the resolution of the two-dimensional amplitude and delay histogram.

The program flow graph in Fig. 5 illustrates the software modules of the algorithm. Once the process is started, the program collects the incoming samples and stores them in a buffer until the desired FFT length is reached. Then the window function is applied and the FFT is calculated. With each new incoming sample two things have to be performed. Before the new sample can be stored in the buffer the old sample has to be shifted into another location of the buffer in order to achieve correct overlapping window functions. In a second step we have to estimate the mixing parameters based on the previous buffer of FFT results. For every incoming input sample we calculate the parameters of one (or two, this depends on the hop size) frequency point of the previous FFT. This is done in the block ”DUETCalc”. When the buffer is filled again, the parameters of all frequency points of the previous FFT have been calculated. Now, the process starts again with the next FFT. For every frequency point the signal power is added to the histogram at the position of the calculated set of parameters by weighting it with the binomial mask. So, the calculation of the histogram is always delayed by one FFT length.

Once every second the histogram is analyzed by the peak detection algorithm within the block ”FindLocMax” which delivers estimates of the mixing parameters of the signal sources. Normally, the peaks in the histogram shift a little bit every second. So, there is a need to decide which peak at time index t_n corresponds to a peak at time index t_{n+1}. This is done by the block ”SortSources”. For every possible permutation of such a peak shifting the squared sum of distances is calculated and the smallest sum

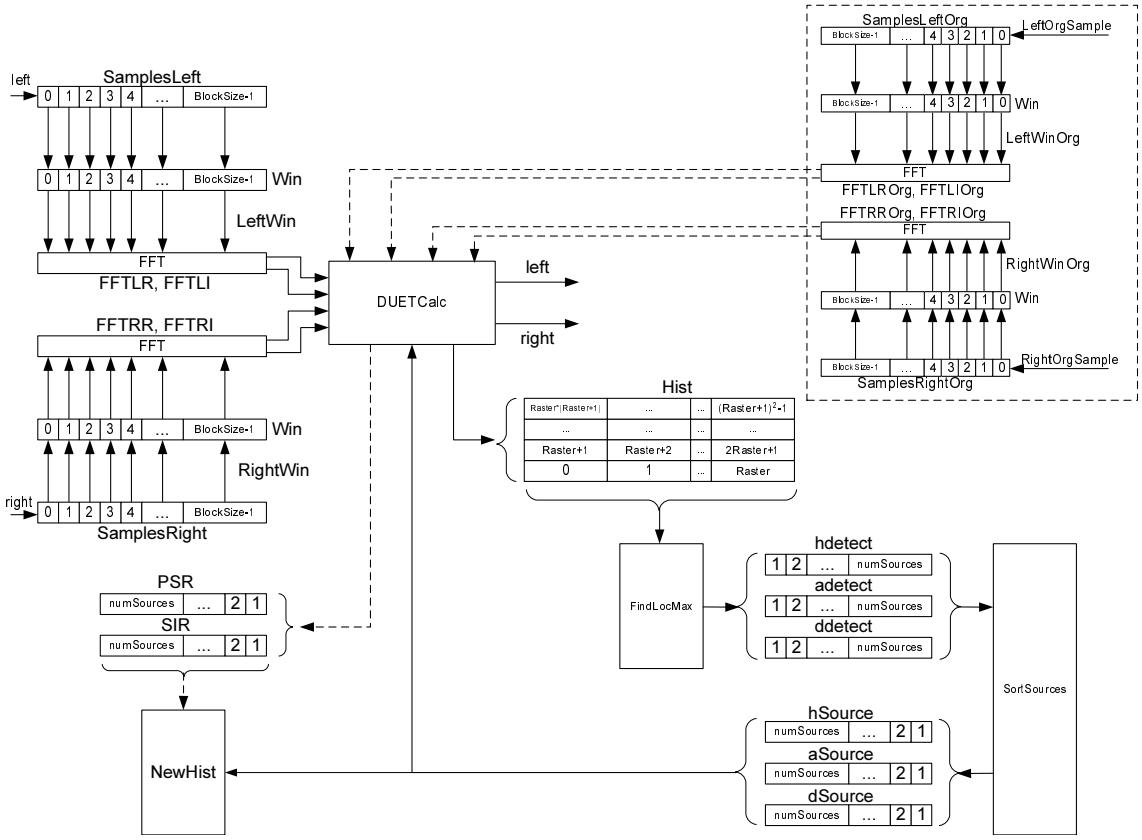

Figure 5: Program flow graph.

represents the most likely permutation. Fig. 6 shows an example for a false and a (most likely) correct shifting relation. The black circles represent the detected peaks at time index t_n and the white circles represent them at time index t_{n+1}. After the first second we estimate the mixing parameters for the first time. These parameters are used for the separation algorithm during the next second. Then, a new set of parameters is estimated, which is used during the third second and so on. So, separation of current samples is always performed based on old parameters. In the worst case, the estimations are one second old. But the results show, that this is acceptable.

The separation algorithm is computed in the block "DUETCalc". Every frequency point of the previous FFT which updates the histogram is also transferred to the separation algorithm. For each frequency point the likelihood function given by Eq. (12) is calculated with the latest set of estimated parameters. The parameters of one source will give the highest likelihood. Then this frequency point is assigned to that source. So, this is a binary assignment, every frequency point is assigned to just one source. For the remaining sources this frequency point is set to zero. Figure 7 shows this procedure for assigning FFT bins $X_1(k,l)$ to the source FFT bins $S_1(k,l),\ldots,S_N(k,l)$. After the IFFT of all $S_1(k,l),\ldots,S_N(k,l)$ and the overlap-add procedure we have all sources $s_1(n),\ldots,s_N(n)$ separated. Obviously, this rough assignment with zero valued FFT bins in between needs further improvements. But this effect is neglected in the actual implementation.

4. PERFORMANCE ANALYSIS

A performance analysis can be done by listening tests which are the most convincing arguments for a special algorithm and objective measurements which are much more difficult to obtain. In the end a good correlation between listening tests and objective measurements should be achieved. The objective measurements are performed by using the original signal sources and their time-frequency representations. The dotted lines in Fig. 5 show the integration of the original source signals and their time-frequency representations into the algorithm for measuring the following performance measures. Following [12] we use three values to measure the performance of the algorithm. First, we define the preserved signal ratio (PSR)

$$\text{PSR}_j = \frac{\sum_{k,l} M_j(k,l)|S_j(k,l)|^2}{\sum_{k,l} |S_j(k,l)|^2} \tag{13}$$

which shows how much power of the original signal is preserved using the time-frequency mask $M(k,l)$. Next, we are interested in how good the mask $M(k,l)$ suppresses interfering sources (Signal to Interference Ratio) calculated by

$$\text{SIR}_j = \frac{\sum_{k,l} M_j(k,l)|S_j(k,l)|^2}{\sum_{k,l} M_j(k,l)|Y_j(k,l)|^2} \tag{14}$$

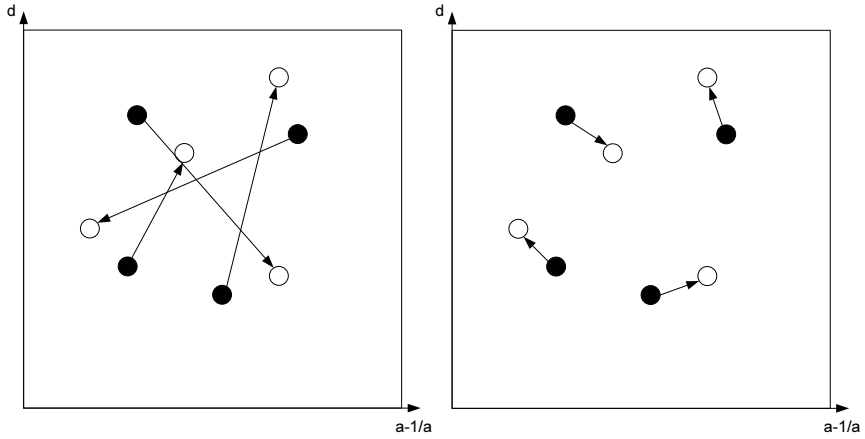

Figure 6: False (left) and correct (right) allocation of the sources.

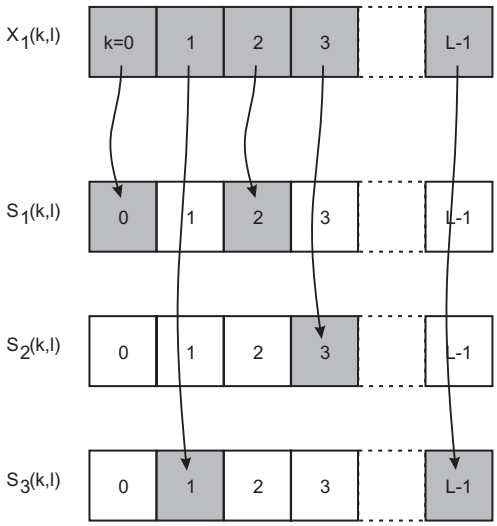

Figure 7: Binary separation decision for three sources.

with

$$Y_j(k,l) = \sum_{i=1;i\neq j}^{N} S_i(k,l) \tag{15}$$

which is the sum of all sources interfering with source j. Combining both values we get a measure for the overall performance which gives us the W-disjoint orthogonality of the sources

$$\mathrm{WDO}_j = \mathrm{PSR}_j - \frac{\mathrm{PSR}_j}{\mathrm{SIR}_j}. \tag{16}$$

We analyzed different mixes with several parameters in order to find an optimal set of parameters (e.q. FFT length) for the best performance. The sampling rate for all mixes was 44.1 kHz. Clearly, artificial mixes based on amplitude and delay panning performed best, because they are produced under anechoic conditions.

Figure 8 shows the averaged W-disjoint orthogonality for different FFT lengths, hop sizes and window functions when two speakers are considered. It is obvious that the hop size and the window function have only little influence on the result. Much more important is the FFT length, leading to results between 0.2 and 0.87, which means the span from not understandable up to nearly perfect results. It should be noted that we prefer to use the greater hop size, because it has a lower computational complexity.

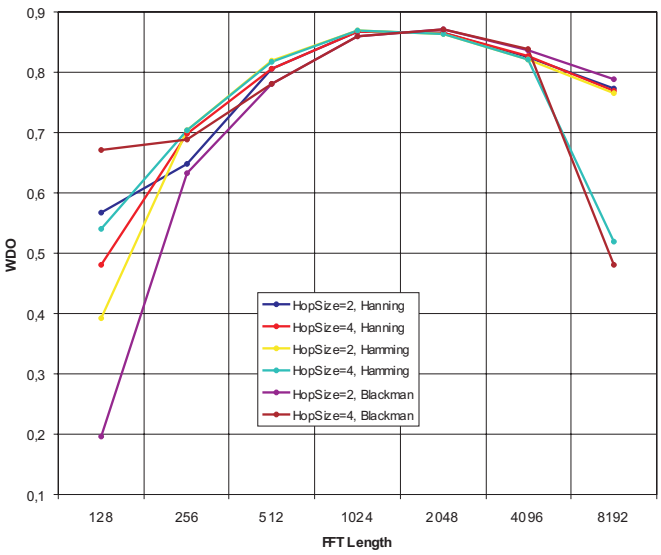

Figure 8: Averaged W-disjoint orthogonality for for different FFT lengths, hop sizes and window functions.

Figure 9 shows the averaged W-disjoint orthogonality for different number of speakers versus the FFT length. As expected, the results are getting worse the more speakers are in the signal mix. Signals with two speakers lead to results above 0.8, which means nearly perfect separation. There are just very few artifacts. The mixes with three speakers are still very good to handle, the W-disjoint orthogonality is above 0.7 what still means there are only some artifacts. Mixes with four speakers are much more complicated. The highest WDO we reached was 0.4, which means that

the separation has many artifacts and can be described as not intelligible. Note that the graph shows the averaged WDO above all four speakers. The speakers were artificially arranged on a diagonal line in front of the microphones. The two speakers in the middle lead to a WDO of around 0.2 while the two speakers at the border have a WDO of around 0.6, which is still intelligible.

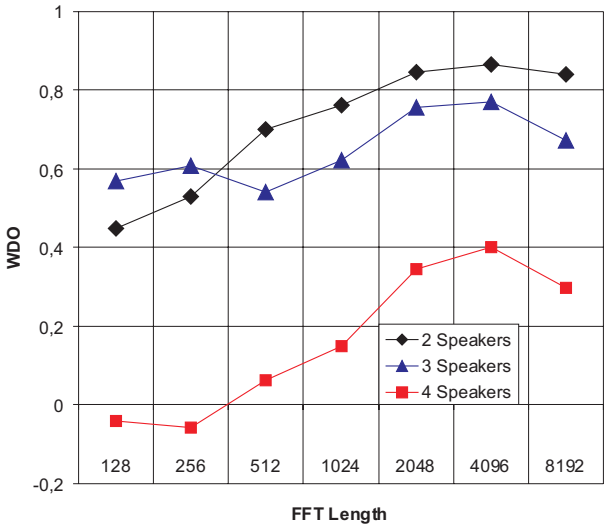

Figure 9: Averaged W-disjoint orthogonality for different number of speakers.

Two-microphone signals recorded in real rooms are still very hard to handle, because one source has not exactly one set of mixing parameters a_j and d_j and the W-disjoint orthogonality condition is not fulfilled, which is also reported in [17]. Nevertheless, the improvements in automatic peak detection and demixing have shown intelligible results. Our analysis shows that recordings in real rooms need more fine-tuning within the parameters to get the best result. The analysis parameters are strongly depending on the incoming signals and have to be adapted to the situation. Further processing steps are necessary to achieve an estimation of how each point on the time-frequency grid is effected by the dominant source signal and the other sources in the signal mix.

5. CONCLUSIONS

A real-time implementation of a source separation algorithm has been introduced. Based on this implementation some test signals have been analyzed. The achieved results show, that the automatic peak detection, the histogram smoothing and the calculation of the maximum likelihood have significantly improved the audio quality. Up to three speakers can be separated nearly perfect. The influence of some program parameter settings have been analyzed. Some improvements still have to be done, for example the extension of the algorithm for moving sources. Also, the number of sources have to be predefined by the user but should be automatically detected. In the case of source signals with room effects, there is still a lot of work to be done, in order to achieve an implementation that finds the best parameters without manual adjustment. Further work will concentrate on dereverberation techniques which can be applied to reduce the influence of room impulse responses. In order to improve the quality of the extracted source signal, several techniques can be incorporated for spectrum estimation of missing frequency bins.

6. REFERENCES

[1] J. Blauert, *Spatial Hearing*, MIT Press, Rev. edition, 1996.

[2] B. Kollmeier, "Cocktail-Partys und Hörgeräte: Biophysik des Gehörs," *Physik Journal*, vol. 1, no. 4, pp. 39–45, 2002.

[3] C. Kyriakakis, P. Tsakalides, and T. Holman, "Surrounded by Sound – Acquisition and Rendering Methods for Immersive Audio," *IEEE Signal Processing Magazine*, pp. 55–66, January 1999.

[4] K. Torkkola, "Blind Separation for Audio Signals – Are we There Yet?," in *Proc. Workshop on Independent Component Analysis and Blind Signal Separation*, Aussois, France, Jan 11-15 1999, pp. 1–6.

[5] J. Anemüller, *Convolutive Blind Source Separation*, Ph.D. thesis, University of Oldenburg, Germany, 2001.

[6] T. Wittkop, *Two-channel Noise Reduction Algorithms motivated by Models of Binaural Interaction*, Ph.D. thesis, University of Oldenburg, Germany, 2001.

[7] H. Viste and G. Evangelista, "An Extension For Source Separation Techniques Avoiding Beats," in *Proceedings DAFx-02*, 2002, pp. 71–75.

[8] C.H. Knapp and G.C. Carter, "The Generalized Correlation Method for Estimation of Time Delay," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. 24, pp. 320–327, 1976.

[9] J. Anemüller and T. Gramß, "Blinde akustische Quellentrennung im Frequenzbereich," in *Fortschritte der Akustik - DAGA '98*, Oldenburg, Germany, 1998, pp. 350–351.

[10] A. Jourjine, S. Rickard, and O.Yilmaz, "Blind Separation of Disjoint Orthogonal Signals: Demixing N Sources from 2 Mixtures," in *ICASSP-2000, Istanbul, Turkey, Vol. 5*, 2000, pp. 2985–88.

[11] S. Rickard and O. Yilmaz, "On the Approximate W-Disjoint Orthogonality of Speech," in *ICASSP-2002, Orlando, USA, Vol. 1*, 2002, pp. 529–532.

[12] O. Yilmaz and S. Rickard, "Blind Separation of Speech Mixtures via Time-Frequency Masking," Submitted to the IEEE Transactions on Signal Processing, November 4, 2002.

[13] D. Arfib, F. Keiler, and U. Zölzer, "Time-frequency Processing," in *DAFX–Digital Audio Effects*, U. Zölzer, Ed., pp. 237–263. J. Wiley & Sons, 2002.

[14] M. Baeck, "Implementierung und Untersuchung eines Verfahrens zur Separation von Audiosignalen aus einem Signalgemisch," Studienarbeit UniBw Hamburg, June 2002.

[15] M. Baeck and U. Zölzer, "Performance Analysis of a Source Separation Algorithm," in *Proceedings DAFx-02*, Hamburg, Germany, 2002, pp. 207–210.

[16] M. Baeck, "Echtzeitimplementierung eines Quellenseparationsalgorithmus für stereophone Audiosignale," Diplomarbeit UniBw Hamburg, April 2003.

[17] V. Hohmann, J. Nix, G. Grimm, and T. Wittkop, "Binaural Noise Reduction For Hearing Aids," in *ICASSP-2002, Orlando, USA, Vol. IV*, 2002, pp. 4000–4003.