

Synchronization of Acoustic Sensors for Distributed Ad-hoc Audio Networks and its Use for Blind Source Separation

Stefan Wehr ^{*}, Igor Kozintsev [†], Rainer Lienhart [†], Walter Kellermann ^{*}

^{*} University Erlangen-Nürnberg
Multimedia Communications and Signal Processing
Cauerstraße 7, 91058 Erlangen, Germany
{Wehr,WK}@LNT.de

[†] Intel Corporation
Microprocessor Research, Intel Labs
3600 Juliette Lane, Santa Clara, CA 95052, USA
{Igor.V.Kozintsev,Rainer.Lienhart}@Intel.com

Abstract

Advanced computing devices equipped with various wired and wireless network capabilities, built-in microphones and audio capture devices are becoming increasingly popular. At the same time, sophisticated signal processing algorithms for hands-free acoustic human-machine interfaces are being developed. Those algorithms are currently restricted to dedicated audio hardware, in part because they require perfectly synchronized audio data. Naive attempts to use the available audio devices for microphone array processing in a distributed wireless setting fails due to the algorithms' sensitivity to deviations in the sampling rates of the distributed devices. We propose here a synchronization scheme, which combines the microphones of different spatially distributed computing devices to an acoustic ad-hoc network. The proposed scheme is capable to significantly compensate the sampling rate deviations of the different audio capture devices and we will show that, as an example, blind source separation performs well with the synchronized data of distributed acoustic sensors.

1. Introduction

Personal computing devices (PCDs) have become omnipresent. Cellular phones, personal digital assistants (PDAs), and laptops are widely used in the daily life of business as well as private people. Those devices are increasingly equipped with many kinds of wireless interfaces like WLAN or bluetooth. Furthermore, modern PCDs are usually equipped with acoustic sensors and audio capture devices which allow, for instance, recording and sending short notes, or making annotations to presentations.

Research in audio signal processing has developed very promising algorithms for hands-free human-machine inter-

faces, e.g., beamforming, acoustic echo cancellation, source separation, and source localization [1, 2]. All these algorithms require perfectly synchronized acoustic sensors. Therefore, dedicated audio hardware is usually provided for those applications: Microphone arrays are permanently mounted in the room, individual microphones are placed in front of each seat, or people wear close-talking microphone (e.g., headsets). All these solutions allow hands-free communication but they are usually expensive and inconvenient. In this paper, we focus on the applications of blind source separation (BSS) for teleconferencing systems based on mobile computing devices. In teleconferences, groups of conference attendants are linked by audio-visual channels. Preferably, only one single person is talking. But actually, especially during discussions multiple people might talk simultaneously. In this case, the far-end groups receive the mixture of reverberated speakers. Focusing on one speaker is difficult and important arguments, ideas, and contributions might be missed. Blind source separation allows extracting the individual speakers from the mixed signals. The far-end side may decide which speaker should be dominant. Alternatively, all speakers might be reproduced individually on the far-end side by spatially distributed loudspeakers. This spatial information supports the audience to follow all speakers and focus on the desired one.

In this paper, we demonstrate how the acoustic sensors of distributed PCDs can be merged to a distributed acoustic sensor array. We will briefly introduce the implemented blind source separation algorithm and we will illustrate the sensitivity of blind source separation to deviations in the sampling rates of microphone signals. We will describe the proposed synchronization scheme and we will discuss our choice of synchronization signal. Finally, the capability of the proposed scheme to synchronize the captured audio data to both a common sampling rate and a common time will be investigated.

2. Related Work

In previous works, we analyzed the sensitivity of microphone array algorithms to sampling rate deviations and we proposed a synchronization scheme to overcome the synchronization problem of distributed audio capture devices [3,4]. We there only focused on measuring and compensating the relative sampling rate deviation of the distributed capture devices with respect to a reference device. Those synchronization schemes basically maximized the cross-correlation between the given and the captured synchronization signal. In this paper, we introduce a new synchronization concept, which is capable to precisely compensate the sampling rate deviation and which, in contrast to previous approaches, synchronizes the captured audio data to a common time.

The problem of time synchronization in distributed multi-process systems has been discussed extensively in the literature in the context of maintaining clock synchrony throughout large geographic areas. Each process exchanges messages with its peers to determine a common clock. Algorithms are explicitly designed to work in the presence of arbitrary clock or process failures. Seminal works have been reported in [5] and [6]. However, the results provided there can not be applied directly to our problem, since the precision of time synchronization depends on the jitter in the duration of the message exchange and clock reading/writing time. NTP, the Network Time Protocol, currently used worldwide for clock synchronization in the best case achieves synchronization errors in the tens of milliseconds - 3 orders of magnitudes more than allowed for our application scenario.

GPS provides a much higher clock resolution. Its reported time is steered to stay always within one microsecond of UTC (Coordinated Universal Time). In practice, it has been within 50 nanoseconds [7]. With the Standard Positioning Service (SPS) a GPS receiver can obtain a time transfer accuracy to UTC within 340 nanoseconds (95% interval). GPS, however, only works outdoors and thus does not completely fit our application scenario.

There is also some recent work on synchronization in wireless sensor networks. In [8], the reference-broadcast synchronization method is introduced. In this scheme, nodes send reference beacons to their neighbors based on a physical broadcast medium. All nodes record the local time at which they receive the broadcasts (e.g., by using the RDTSC (read-time stamp counter) instruction of the Pentium processor family). Based on the exchange of this information, nodes can convert their local clock time into each others clock time. Although promising, the worst case clock deviation of 150 microseconds reported in [9] is too high for our application scenario. Also, it requires significant software changes at the driver level in each platform, while our

system can work with any legacy hardware with multiple audio channels. It is also not clear which precision can be achieved on heterogeneous platforms where delay and jitter can vary significantly between the various platforms.

3. Frequency-domain blind source separation

3.1. The BSS application

Blind source separation is motivated by the so-called *cocktail party problem*: Multiple speakers are talking simultaneously in a reverberant acoustic environment. Focusing on the desired speaker, the "speaker of interest", is quite challenging for the participants, even if they are on site and if they have the opportunity to track the speaker and exploit spatial information, e.g., by turning the head or by looking straight to the speaker. Hearing impaired people, who can only rely on the output signal of their hearing aids, often have problems to focus on one speaker. Speech recognition systems are severely affected by interfering speakers. The intelligibility of the far-end speaker in teleconferences and in hands-free mobile communication is significantly degraded by interfering speakers. Several approaches to the *cocktail party problem* were proposed in the past, popularizing the terms "blind source separation" and "independent component analysis". In this paper, we refer to the approach of C. Fancourt and L. Parra [10].

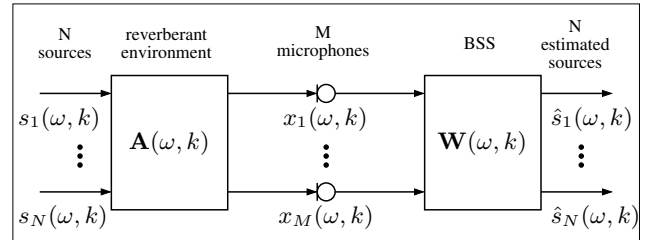


Figure 1. Block diagram of the BSS problem for N sources and M sensors.

Figure 1 illustrates the BSS problem. ω indexes the discrete Fourier transform (DFT) frequency bins and k is the discrete time index. Matrix $\mathbf{A}(\omega, k)$ represents the propagation of the N sources $s_i(k)$, $i = 1, \dots, N$, in the acoustic reverberant environment to the M microphones. The microphone signals are denoted as $x_j(\omega, k)$, where $j = 1, \dots, M$. The BSS algorithm computes the unmixing matrix $\mathbf{W}(\omega, k)$ in the DFT domain and returns the N estimated source signals $\hat{s}_i(\omega, k)$, $i = 1, \dots, N$. The discrete time index k suggests that matrix $\mathbf{A}(\omega, k)$ is time-variant. Actually, we assume only slow changes in $\mathbf{A}(\omega, k)$ with respect to the adaptation of the unmixing matrix $\mathbf{W}(\omega, k)$. In vector notation, the BSS scenario in Figure 1 may be written in the

DFT-domain as

$$\mathbf{x}(\omega, k) = \mathbf{A}(\omega, k) \cdot \mathbf{s}(\omega, k) \quad (1)$$

$$\hat{\mathbf{s}}(\omega, k) = \mathbf{W}(\omega, k) \cdot \mathbf{x}(\omega, k) \quad (2)$$

Note that the convolutions of the source signals with the room impulse responses become multiplications in the DFT-domain. For further details on this algorithms, please refer to [10].

The separation performance of the BSS algorithm in dB is given by the improvement of the signal-to-interference ratios (SIRs) between the BSS-input and the BSS-output in each channel i :

$$\text{Gain}_i(k) = \text{SIR}_{\text{out},i} - \text{SIR}_{\text{in},i} \quad (3)$$

Unlike [10, 11] we compute the signal-to-interference ratios in the time-domain as follows:

$$\text{SIR}_{\text{in},i}(k) = 10 \log_{10} \left[\frac{\text{cov} \left\{ x_{i,i}(k) \right\}}{\text{cov} \left\{ \sum_{l \neq i} x_{i,l}(k) \right\}} \right] \quad (4)$$

$$\text{SIR}_{\text{out},i}(k) = 10 \log_{10} \left[\frac{\text{cov} \left\{ \sum_j \hat{s}_{i,j,i}(k) \right\}}{\text{cov} \left\{ \sum_{l \neq i} \sum_j \hat{s}_{i,j,l}(k) \right\}} \right] \quad (5)$$

$x_{j,i}(k) = a_{ji}(k) * s_i(k)$ represents the contribution of the source signal $s_i(k)$ to the microphone signal $x_j(k)$ and $\hat{s}_{l,j,i}(k) = w_{lj}(k) * a_{ji}(k) * s_i(k)$ represents the contribution of the source signal $s_i(k)$ to estimated source signal \hat{s}_l via microphone j . Thus, the desired signal in channel i appears in the numerator and all interfering signals are summarized in the denominator. With this definition of the SIRs, we are able to represent the case of absent interfering speakers by an infinite SIR.

3.2. Sensitivity of BSS to sampling rate deviations

Microphone array algorithms mentioned in Section 1 implicitly assume perfectly synchronized sensor data, i.e. the sampling rate of all audio capturing devices consistently match. This is guaranteed by multi-channel input devices operating at a specific sampling rate. Exemplarily for all microphone array processing algorithms, we investigated the sensitivity of the BSS algorithm introduced in section 3.1. Two source signals were played back in a low-reverberation chamber and recorded by two perfectly synchronized microphones. The reverberation time was $T_{60} = 50\text{ms}$ and the signal duration was 23 seconds. With respect to the original sampling rate of 16kHz, one sensor signal was resampled in steps of 0.1Hz. The underlying resampling algorithm [12, 13] was also used for resampling the captured

audio data in the proposed synchronization scheme. For each resampled sensor signal, we computed the time- and channel-averaged gain of BSS.

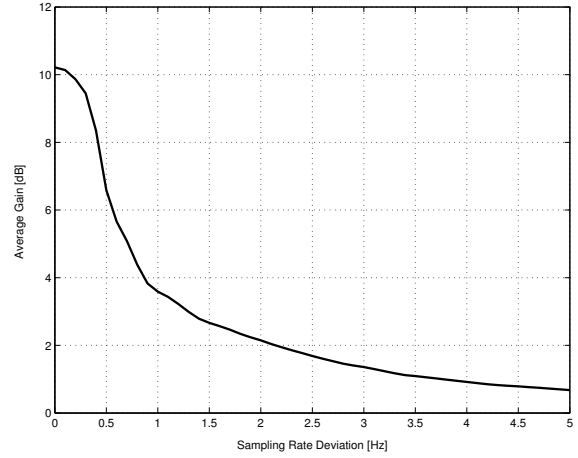


Figure 2. Sensitivity of BSS to sampling rate deviations in a low-echoic chamber with reverberation time $T_{60} = 50\text{ms}$, a signal duration of 23 seconds, the sampling rate 16kHz, and a resolution of 0.1Hz.

Figure 2 shows the impact of the resampling on the separation performance. We see that the investigated BSS algorithm is extremely sensitive to sampling rate deviations. A deviation of only a few Hertz already degrades the algorithm to a level, where no separation performance is achievable. Therefore, the sensor signals of distributed capture devices necessarily have to be synchronized to a common sampling rate with an accuracy of a fraction of 1 Hertz.

4. Synchronized distributed acoustic sensors

In this section, we propose our synchronization scheme. Firstly, we define the setup for distributed acoustic sensor networks and we discuss different methods for transmitting the synchronization signal. Secondly, we consider several potential synchronization signals. Finally, we propose a scheme that synchronizes the incoming audio data of all PCDs both to a reference sampling rate and to a reference time.

4.1. Setup for distributed acoustic sensor networks

Figure 3 shows the setup for P distributed PCDs. Note that every capturing device is considered as an individual PCD, although it could be mounted in the same computer. Every capturing device is equipped with an individual clock, which is (in common audio devices) not accessible by the user or by other devices in the ad-hoc network. In order to capture the (relative) sampling rate deviation between the clock of the BSS platform and the clocks of the

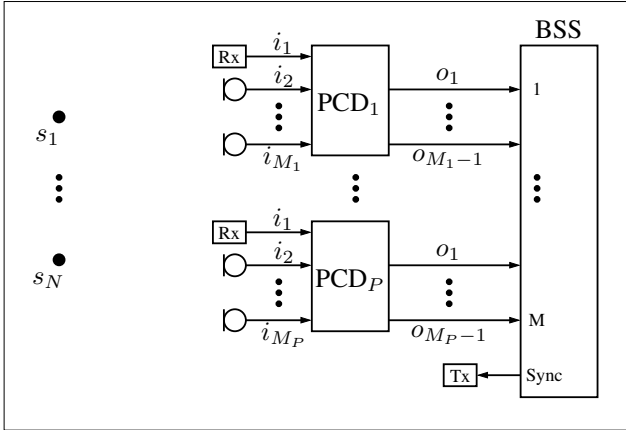


Figure 3. Setup of a distributed acoustic sensor network consisting of P spatially distributed PCDs and one BSS-performing platform with M acoustic sensor signals.

capturing devices, we send a digital synchronization signal through one of the D/A-converters of the BSS platform and we re-digitize it with an A/D-converter in each of the capturing devices. Therefore, every audio capturing device PCD_i has to spare one of its M_i input audio channels. We might consider to simply transmit the synchronization signal through the air and capture it with acoustic sensors. This approach has many drawbacks. Firstly, the users of the distributed sensor array would most likely not accept any audible synchronization signals. Therefore, the synchronization signal had to be below the auditory threshold, where the signal would severely be affected by interferences. Secondly, the synchronization signals would interfere with the speaker signals, leading to a degradation of the recording quality. Thirdly, the captured audio data has to be synchronized to a common time. This ideally requires simultaneous arrival of the synchronization signal at all involved PCDs. That would not be feasible due to different propagation times. In our experiments, we used the U.S. Robotics SoundLink system [14] in order to wirelessly transmit the synchronization signal by radio-frequency channels. The transmitter modulates the baseband synchronization signal at the sync output of the BSS platform to the carrier frequency of the wireless audio delivery system. The receivers demodulate the FM signal and provide the audio capture devices with the baseband synchronization signal. The propagation time of the synchronization signal is negligible and simultaneous arrival can be assumed.

4.2. Synchronization signal candidates

We now illustrate measuring the *relative* sampling rate deviation between the audio devices in the BSS platform and in the PCDs. Note that knowing the *absolute* sampling

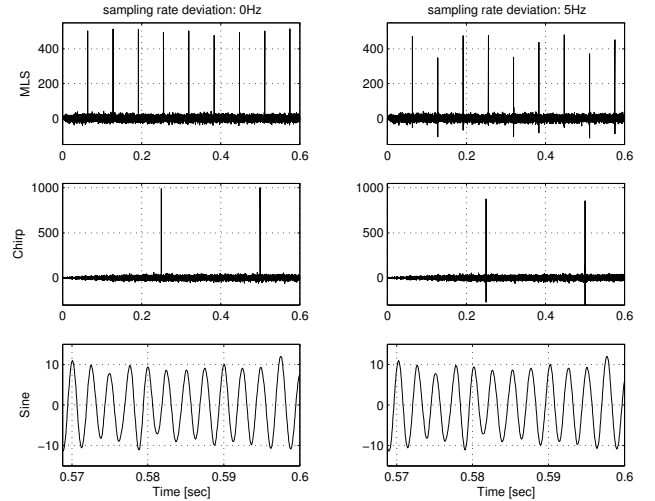


Figure 4. Analysis of the potential synchronization signals: Cross-correlations between one period of the known reference signals and the captured synchronization signals for perfectly synchronized clocks (left) and for 5Hz sampling rate deviation (right) for an AWGN-channel.

rates is not necessary for synchronizing the captured audio data to a common sampling rate. In the time-domain, we may examine the cross-correlation between the captured (i.e. digitized) synchronization signal and one period of the reference signal. The samples of the reference signal are known and thus available in all PCDs. We investigated several synchronization signals: a Maximum Length Sequence (MLS) [15] of order 10, a chirp signal that swept four times a second from 0Hz to 8kHz, and a sinusoidal signal with frequency 400Hz.

Figure 4 shows the cross-correlations between one period of the (known) reference signals and the corresponding captured synchronization signals in the PCD for an additive white Gaussian noise (AWGN) channel. The assumed sampling rate of the reference signal is 16kHz. On the left hand side, the PCD is perfectly synchronized to the BSS platform. On the right hand side, we simulate an unsynchronized PCD by increasing the sampling rate of the captured synchronization signal by 5Hz. Note that we still observe cross-correlation peaks in case of deviating clock rates. Basically, we may scale the synchronization signal at the sync output in order to obtain a) equal magnitudes, or b) equal powers and thus equal signal-to-noise ratios (SNRs). The best SNR, and thus the best achievable robustness to AWGN, is obtained by the highest possible magnitude. The upper bound of the signal magnitude is physically given by the limiters in the A/D-converters of the capturing devices and in the D/A-converter in the BSS platform. For best robustness to AWGN, we will always try to approach this physical upper bound without introducing

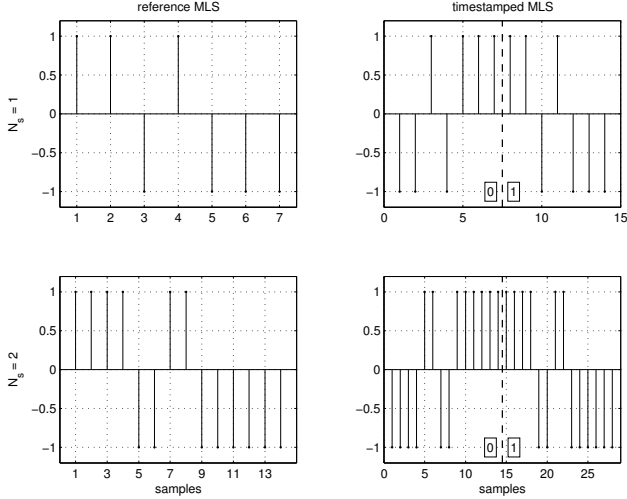


Figure 5. Reference MLS (left) and MLS timestamped with bits 0 and 1 (right). The MLS order is 3 and the number of samples per MLS symbol N_s is 1 and 2 respectively.

nonlinearities due to clipping. Therefore, we investigated case a) in Figure 4 and the SNRs are: $\text{SNR}_{\text{MLS}} = 2.92\text{dB}$, $\text{SNR}_{\text{Chirp}} = -0.03\text{dB}$, and $\text{SNR}_{\text{Sinusoidal}} = -0.20\text{dB}$.

Due to the maximum peak-to-average power ratio (PAR) for a given magnitude, the MLS is optimum in terms of robustness to distortion and AWGN. Due to the smaller period of the MLS compared to the wideband chirp signal, the MLS provides a higher density of cross-correlation peaks, which will be used for measuring the sampling rate deviation later. The choice of synchronization signal may mainly be affected by the users' preferences, but we focus on the MLS in this paper.

For the MLS of order 10 (Fig. 4), we illustrate measurement of the relative sampling rate deviation. The period of this MLS is 1023 samples and thus, in case of perfect synchronization, the cross-correlation peaks in Figure 4 (left) are 1023 samples apart. Note that this distance is fixed for all sampling rates, as long as the sampling rates of the BSS platform and the PCD exactly match. If the sampling rate of the PCD exceeds the sampling rate of the BSS platform, one captured MLS period in the PCD consists of $1023 + x$ samples ($x > 0$). Then, cross-correlating the 1023 samples of the reference MLS with the $1023 + x$ samples of the captured MLS results in a cross-correlation peak at sample position $1023 + x$. Accordingly, the cross-correlation peak is located at sample position $1023 + x$ ($x < 0$) for a PCD sampling rate undershooting the BSS platform sampling rate. Assuming constant sampling rates, the distance between two adjacent cross-correlation peaks in the PCD equals $1023 + x$ and computing x yields the relative sampling rate deviation. Note that computing the relative sampling rate deviation is not restricted to two adjacent peaks,

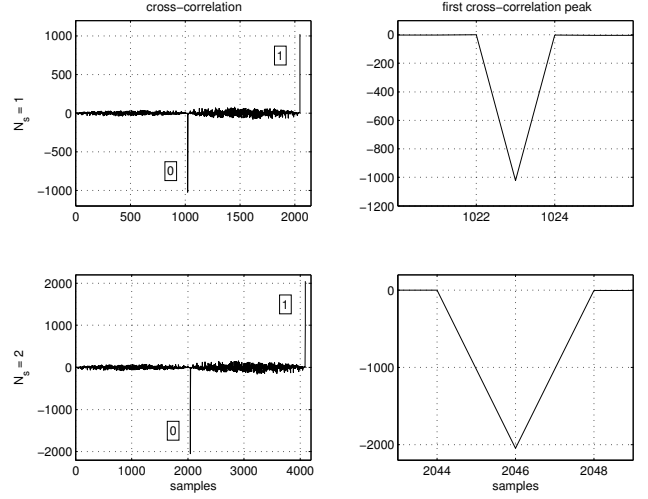


Figure 6. Cross-correlations between the reference MLS and the corresponding timestamped MLS depicted in Fig. 5 (left) and cross-correlation peaks corresponding to timestamp bit 0 (right). The MLS order is 10 and the number of samples per MLS symbol N_s is 1 and 2 respectively.

it may actually be performed with two far-distant peaks if the number of enclosed periods is counted.

Now, we illustrate carrying timestamps with the MLS. Generally, the word length of the binary timestamp is N_b bits. On the left hand side of Figure 5, we see one period of the MLS of order 3. The right hand side shows the MLS carrying the timestamp "01" for $N_b = 2$. Modulating the reference MLS by bit 0 (left of dashed line) corresponds to a multiplication with -1, a modulation by bit 1 (right of dashed line) does not change the reference MLS. N_s denotes the number of samples per MLS symbol. For $N_s = 2$, every MLS symbol is represented by two identical samples and the MLS period is accordingly increased by the factor 2. Figure 6 shows the effect of N_s on the cross-correlation peaks. Higher N_s values result in wider and higher peaks, which are (due to the increased period) further apart. This effect supports both detecting the peaks and interpolating fractional peak locations. We thus may increase the accuracy of measuring the relative sampling rate deviation. Figure 6 also illustrates the timestamp detection. The peak sign corresponds to the modulation factor (-1 or +1) and we thus obtain the timestamp bits. Due to phase shifts in the channel (including the pre-filters in the A/D-converter), the peak signs might be swapped. By using a sufficiently high word length N_b (e.g. $N_b = 10$) and by deterministically sequencing the timestamps, we are able to detect and compensate those phase shifts and we are able to detect missing timestamps in the receivers. Further refinements, e.g. maximizing the Hamming distance between two subsequent timestamps, may be considered.

4.3. Synchronization scheme

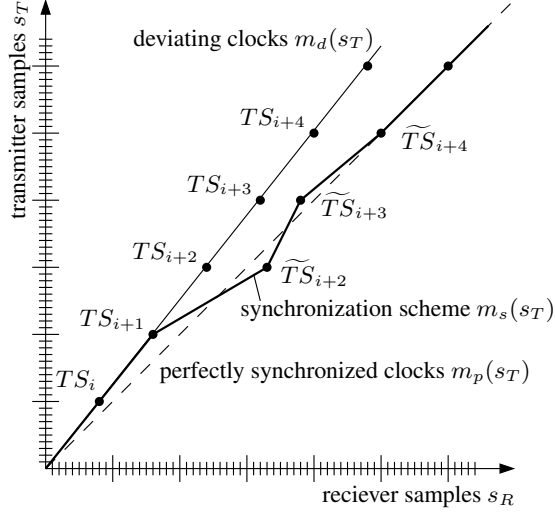


Figure 7. Adaptive compensation of the clock rate deviation between the transmitting device and the receiving device

Figure 7 illustrates the adaptive compensation of deviating clock rates. The major ticks mark the reference sample positions of the timestamps TS in the transmitter and in the receiver respectively. More precisely, they mark the sample positions of the cross-correlation peaks corresponding to the last bit of every timestamp (Fig. 6). $m_p(s_T)$, $m_d(s_T)$, and $m_s(s_T)$ denote the (for $m_s(s_T)$ only piecewise) linear mapping functions for perfectly synchronized clocks (dashed solid straight line), for deviating clocks (solid straight line), and for the proposed synchronization scheme (bold solid line), which map the transmitter samples s_T to the receiver samples s_R .

$$s_R = m_p(s_T) = a_p s_T + b_p \quad (6)$$

$$s_R = m_d(s_T) = a_d s_T + b_d \quad (7)$$

$$s_R = m_s(s_T) = a_s s_T + b_s \quad (8)$$

In this example, the offsets b_p and b_d equal zero. Note that all gradients a have to be positive. In some applications of the proposed synchronization scheme, further constraints to the gradient a_s may be required. In the case of perfectly synchronized transmitter and receiver clocks, the detected timestamps in the receiver are exactly mapped to the reference sample positions, but for deviating clock rates the timestamps' sample positions differ from the reference positions. In the depicted scenario, the transmitter's clock T_T runs faster than the receiver's clock T_R resulting in a relatively higher sampling rate of the transmitter. In order to approach perfectly synchronized clocks, we exploit both the time information given by the timestamps TS and the measured sampling rate deviation given

by the timestamps' sample positions in the receiver (see section 4.2). We might consider to just resample the captured audio data to the transmitter's sampling rate. This would compensate the relative sampling rate deviation ($a_s \approx a_p$) but a constant sample offset would be persistent. Instead, our proposed synchronization scheme approaches perfectly synchronized clocks: Based on the sample positions of the timestamps TS_i and TS_{i+1} in the receiver, we extrapolate the sample position of timestamp TS_{i+2} in the receiver. By mapping TS_{i+2} parallel to the abscissa, we obtain \tilde{TS}_{i+2} . With ideally computed a_s and b_s , \tilde{TS}_{i+2} would exactly hit the dashed line representing perfectly synchronized clocks. Due to estimation errors and other deviations, \tilde{TS}_{i+2} only approaches the reference sample position on the dashed line. We observed that, even for the first iteration, \tilde{TS}_{i+2} approaches the dashed line extremely close, comparable to \tilde{TS}_{i+4} . (For illustrating the convergence to perfectly synchronized clocks and for improving the clearness of the plot, we exaggerated the approximation error of the first two iterations \tilde{TS}_{i+2} and \tilde{TS}_{i+3} .) For each detected timestamp, we re-compute a_s and b_s and thus the segments of $m_s(s_T)$ approach $m_p(s_T)$. Due to (long-term) time-variant clocks and estimation errors, the gradients of $m_s(s_T)$ and $m_p(s_T)$ usually differ ($a_s \neq a_p$) and, if timestamp detection fails, the synchronization would therefore degrade. In this case of missing timestamps in the receiver, we switch to the long-term estimation of the sampling rate deviation ($a_s \approx a_p$), resulting in a minor sample offset of the synchronized signal ($b_s \neq b_p$), where the sampling rate deviation is still compensated. When detecting a new timestamp, we return to the abovementioned synchronization scheme. The scenario of missing timestamps is not covered in Figure 7.

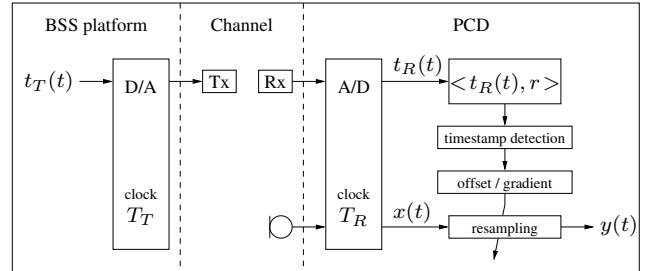


Figure 8. Proposed synchronization scheme for one PCD with two audio input channels and wireless transmission of the synchronization signal.

Figure 8 illustrates the derived synchronization scheme for one PCD with two audio input channels. $t_T(t)$ and $t_R(t)$ are the discrete timestamped synchronization signals in the transmitting and in the receiving device, respectively. $t_T(t)$ is a timestamped version of the reference signal r . Note that the reference signal is time-invariant and determinis-

tic and therefore known in all PCDs. The internal clocks of the transmitting device and the receiving device have the periods T_T and T_R , which may be time-variant. The A/D-converter of the PCD digitizes both the received synchronization signal and the time-continuous microphone signal with the sampling rate $1/T_R$. The block $\langle t_R(t), r \rangle$ computes the cross-correlation between the (windowed) signal $t_R(t)$ and the reference signal r . The subsequent timestamp detection analyzes the cross-correlation for the significant peaks that we have seen in Figure 4 and extracts the timestamps. Based on the position of the cross-correlation peaks and on the time information, we compute the above-mentioned offset and gradient of the linear mapping function and we accordingly adapt the resampling unit. The resampled audio data of all PCDs (in this example signal $y(t)$) are now synchronized both to the sampling rate and to the common time of the BSS platform.

5. Experiments

For our simulations, we implemented the approach of C. Fancourt and L. Parra [10] in Matlab. The parameters were optimized for 16kHz sampling rate. All recordings were conducted at 44.1kHz sampling rate and they were downsampled to 16kHz.

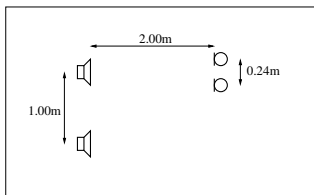


Figure 9. Simulation setup in the low-echoic chamber ($T_{60} = 50\text{ms}$) with two acoustic sensors and two source signals.

Figure 9 shows the simulation setup in the low-reverberation chamber ($T_{60} = 50\text{ms}$). The proposed synchronization scheme is capable to synchronize the audio input channels of P spatially distributed PCDs as depicted in Figure 3. However, we only consider two PCDs with two audio input channels each in our experiments. The two sensors were 24cm apart and the two sources signals, a male and a female speaker, were played back with two loudspeakers. One loudspeaker was positioned in the broadside direction of the two-sensor-array at a distance of 2.00m. The other loudspeaker was located 1.00m off the broadside direction, the distance to the axis of the microphone array remained 2.00m.

For the recordings of about 60 seconds, we used two stereo soundcards: An on-board soundcard (Realtek AC97 Audio) and a PCI soundcard (SoundBlaster Live), both installed in one desktop computer. For playing back the two source signals, we used an external USB audio device (M-

Audio Quattro) that was connected to a laptop (Dell Latitude D600). The internal soundcard (SigmaTel C-Major Audio) of this laptop was playing back the synchronization signal.

The investigated scenarios are:

1. Two distributed unsynchronized devices

In this scenario, the first microphone signal was recorded with one channel of the SoundBlaster Live and the second microphone signals was recorded with one channel of the Realtek AC97 Audio. The second channels of both devices were unused. This scenario represents a general distributed sensor array with different sampling rates for each microphone signal.

2. Two distributed synchronized devices

Additionally to scenario 1, we recorded the synchronization signal with the second channel of the two capture devices. We implemented an audio capture software, which is based on the proposed synchronization scheme. This application synchronizes the captured audio data to the MLS-playing device and it writes the synchronized data to audio files. The audio capture application may be requested from Intel [16].

3. One perfectly synchronized device

The two sensor signals were recorded with the two input channels of one device. Since the two microphone signals are digitized by the same clock, the recordings are perfectly synchronized. The separation performance for these recordings marks the upper bound of the incorporated BSS algorithm in this acoustic environment for this setup. For the recordings with the SoundBlaster Live and for the recordings with the Realtek AC97 Audio, we obtained almost the same separation performance, where the difference is negligible (see Table 1). For clarity, we only show the separation performance for one of the perfectly synchronized devices in Figure 10.

Table 1. Separation performance of the BSS algorithm, averaged over both the time and the channels.

Investigated scenario	Gain
One perfectly synchronized device	16.9 dB
SB Live	16.7 dB
Realtek	16.7 dB
Two distributed synchronized devices	11.1 dB
Two distributed unsynchronized devices	-0.4 dB

Figure 10 shows the separation performance of the BSS algorithm, averaged over the two BSS-channels, for the three investigated scenarios. Table 1 summarizes the time- and channel-average of the separation performance of the

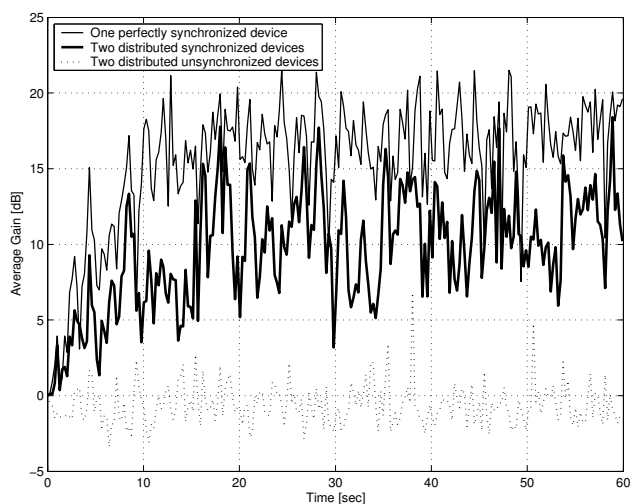


Figure 10. Separation performance of the BSS algorithm, averaged over the channels.

BSS algorithm. The separation performance for the unsynchronized devices, which is even negative in this simulation, demonstrates the necessity of synchronization for distributed acoustic sensor networks.

We observe that the proposed synchronization scheme provides the BSS algorithm with sufficiently precisely synchronized data. We obtain significant separation performance, which is consistent over the simulation period of 60 seconds. Referring to the sensitivity of BSS to sampling rate deviations depicted in Figure 2, the achieved separation performance of BSS indicates high precision of the proposed synchronization scheme.

6. Conclusion

We introduced a synchronization scheme that allows merging distributed audio capture devices to an ad-hoc acoustic sensor network. This allows performing microphone array processing algorithms without being restricted to dedicated audio hardware. We could demonstrate that a blind source separation algorithm, which is extremely sensitive to sampling rate deviations, performs well with the synchronized audio data of our proposed scheme. Existing synchronization methods either do not provide the required accuracy, or they are generally inapplicable for (indoor) microphone array processing. In previous works, we already addressed the problem of compensating the sampling rate deviations. In this paper, we additionally solved the problem of aligning the captured audio data to a common timeline. We investigated several potential synchronization signals and their capability both to carry time information and to accurately capture the sampling rate deviations. This novel synchronization scheme provides microphone array

processing algorithms with temporally consistent and precisely synchronized audio data. As far as we know, the precision of our proposed synchronization scheme is unique in the field of distributed audio signal processing.

References

- [1] M.S. Brandstein and D.B. Ward (eds.), *Microphone Arrays: Signal Processing Techniques and Application*, Springer-Verlag, Berlin, May 2001.
- [2] Y. Huang and J. Benesty (eds.), *Audio Signal Processing For Next-Generation Multimedia Communication Systems*, Kluwer Academic Publishers, Boston, 2004.
- [3] R. Lienhart, I. Kozintsev, S. Wehr, and M. Yeung, "On the importance of exact synchronization for distributed audio signal processing," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, pp. IV – 840–3, April 2003.
- [4] R. Lienhart, I. Kozintsev, and S. Wehr, "Universal synchronization scheme for distributed audio-video capture on heterogeneous computing platforms," *Proceedings of the eleventh ACM international conference on Multimedia*, vol. 4, pp. 263–266, 2003.
- [5] L. Lamport and P.M. Melliar-Smith, "Synchronizing clocks in the presence of faults," *J. ACM*, vol. 32, no. 1, pp. 52–78, 1985.
- [6] D.L. Mills, "Internet time synchronization: the network time protocol," *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482–1493, 1991.
- [7] United States Naval Observatory, "GPS timing data & information," http://tycho.usno.navy.mil/gps_datafiles.html.
- [8] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 147–163, 2002.
- [9] M. Mock, R. Frings, E. Nett, and S. Triakliotis, "Clock synchronization for wireless local area networks," *IEEE 12th Euromicro Conference on Real-Time Systems*, pp. 183–189, 2000.
- [10] C. Fancourt and L. Parra, "The coherence function in blind source separation of convolutive mixtures of non-stationary signals," *IEEE Workshop on Neural Networks for Signal Processing*, pp. 303–312, 2001.
- [11] L. Parra and C. Spence, "On-line blind source separation of non-stationary signals," *Journal of VLSI Signal Processing*, vol. 26, no. 1/2, pp. 39–46, August 2000.
- [12] J.O. Smith and P. Gossett, "A flexible sampling-rate conversion method," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 19.4.1–19.4.2, March 1984.
- [13] J.O. Smith, "Digital audio resampling home page," January 28, 2002, <http://www-ccrma.stanford.edu/~jos/resample/>.
- [14] U.S. Robotics, "Soundlink wireless audio delivery system," <http://www.usr.com/products/device/p-device-product.asp?sku=USR6003>.
- [15] D.D. Rife and J. Vanderkooy, "Transfer-function measurement with maximum-length sequences," *J. Audio Eng. Soc.*, vol. 37, no. 6, pp. 303–312, June 1989.
- [16] Intel Corporation, "DAIO Library v0.5," {Igor.V.Kozintsev,Rainer.Lienhart}@Intel.com.