Abstract—
In this paper, we present the Blind Speech Separation and Dereverberation (BSSD) network, which performs simultaneous speaker separation, dereverberation and speaker identification in a single neural network. Speaker separation is guided by a set of predefined spatial cues. Dereverberation is performed by using neural beamforming, and speaker identification is aided by embedding vectors and triplet mining. We introduce a frequency-domain model which uses complex-valued neural networks, and a time-domain variant which performs beamforming in latent space. Further, we propose a block-online mode to process longer audio recordings, as they occur in meeting scenarios. We evaluate our system in terms of Scale Independent Signal to Distortion Ratio (SI-SDR), Word Error Rate (WER) and Equal Error Rate (EER).

Index Terms—Multi-channel speech separation, beamforming, dereverberation, speaker identification, triplet mining

I. INTRODUCTION

Speaker separation and speech enhancement is of paramount significance in many voice applications, such as handsfree teleconferencing or meeting scenarios. Especially in human-machine interfaces, where high-performance Automatic Speech Recognition (ASR) systems are essential, both speech intelligibility and quality play an important role. Fueled by the success of deep learning, both speaker separation and speech enhancement have made major advances over the last years [1].

When multiple microphones are available, spatial information can be exploited as speaker sources are directional. Mask-based beamforming has been shown to be advantageous for this task [2], [3]. In particular, a neural network is leveraged to estimate a time-frequency mask of the desired signal [4], [5], [6], [7]. This mask is then used to compute the spatial covariance matrices required to construct a frequency-domain beamformer [8]. This approach has been further extended into the domain of complex numbers, where complex-valued neural networks [9] are used to directly estimate complex beamforming weights from noisy observations [10], [11].

Simultaneously to the success of neural beamforming, single-channel speaker separation techniques have also progressed dramatically. Frequency-domain algorithms such as Deep Clustering (DC) [12], Permutation Invariant Training (PIT) [13] and Deep Attractor Network (DAN) [14] rely solely on spectral features. Time-domain algorithms such as Wave-U-Net [15], TasNet [16] and Conv-TasNet [17] delivered promising results.

Recently, some of these single-channel algorithms have been combined with a mask-based beamformer. In particular, a neural network estimates a gain mask of the desired signal, which is then used to construct a frequency-domain beamformer, i.e.: Beam-TasNet [18], SpeakerBeam [19], [20]. Neural Speech Separation [21], Multi-Channel Deep Clustering [22], [23], and Convolutional Beamforming [24]. More recently, end-to-end multi-channel speech separation has been done entirely in time domain. By leveraging spatial cues among the multi-channel signals such as the Inter-channel Time Difference (ITD) or Inter-channel Phase Difference (IPD), the desired signal is estimated directly in time domain [25]. We further extend this approach by addressing the following three issues:

A. Open number of sources
Many source separation algorithms are limited to a predefined number of sources which they can separate [13], [14], [16], [17], [15], [18], [25], [27]. Exceptions are k-means clustering [12] and Recurrent Selective Attention Network (RSAN) [28]. We propose an iterative approach to separate an unknown number of sources, by leveraging the spatial information encoded within the data. As shown in the experiments section, we tested our system with up to four overlapping speakers.

B. Distant speaker separation
While close-talk speech separation models yield impressive performance, far-field speech separation is still a challenging task [29], [30]. Especially in real-world scenarios, reverberation and echoes cannot be ignored, as they severely degrade speech intelligibility and ASR performance [31]. Various deep learning based methods have been proposed for dereverberation [32], [33], [34], most of which are based on the Weighted Prediction Error (WPE) algorithm [35]. As this algorithm is restricted to the frequency domain, we chose a more general approach which learns to separate overlapping speakers from a reverberated mixture in both time- and frequency-domain.

C. Speaker Identification
To be useful in real-world applications, a speaker separation algorithm has to be at least block-online capable, i.e.; a short block of audio is processed at a time. This implies a permutation problem at block level, requiring speaker diarization [36], [37]. Therefore, identifying the separated speakers in each

Lukas Pfeifenberger and Franz Pernkopf are with the Intelligent Systems Group at the Signal Processing and Speech Communication Laboratory, Graz University of Technology, Graz, Austria.

This work was supported by the Austrian Science Fund (FWF) under the project number P27803-N15. Furthermore, we acknowledge NVIDIA for providing GPU computing resources.

Lukas Pfeifenberger and Franz Pernkopf, Senior Member, IEEE
block of audio is necessary. A speaker identification algorithm is agnostic to the spoken text, and only relies on the speaker characteristics embedded in the waveform. Embedding vectors are used to map utterances into a feature space where distances correspond to speaker similarity [36]. Typically, i-Vectors [39] or x-Vectors [40] are used for this task. Algorithms such as Deep Speaker [41] rely on contrastive loss or triplet loss to learn embeddings on a very large set of speakers [42], [43], [44], [45]. We chose the triplet loss as its performance on small batch sizes is advantageous [46].

In this paper, we introduce our Blind Speech Separation and Dereverberation (BSSD) network, which performs separation, dereverberation and speaker identification in a single neural network. The only knowledge required is that of the microphone array geometry. We propose a frequency-domain variant (BSSD-FD), and time-domain variant (BSSD-TD). Our contributions are: Unsupervised speaker localization; separation of each speaker using adaptive beamforming; dereverberation of each source; and speaker diarization using embedding vectors. We evaluate our system in both offline and block-online mode. Further, we report the performance in terms of SI-SDR, WER and EER against similar state-of-the-art algorithms for speaker separation.

II. SYSTEM MODEL

We assume a standard meeting scenario, where multiple speakers may talk simultaneously in an arbitrary room, i.e. an office. The position and number $C$ of the concurrent speakers is unknown. We place a circular microphone array with $M$ microphones in the center of the room, i.e. on a table. Figure 1 provides an example with three speakers. We assume that each speaker has a direct line of sight towards the microphone array, i.e. the speaker is not obscured by a corner, or standing in the next room. Each speaker is assumed to be stationary, except for minor movements. Further, the room may have a significant amount of reverberation. Note that we do not assume any diffuse or directional noise sources in this paper. For speech separation in the presence of ambient noise, we refer the interested reader to [7].

![Fig. 1: Meeting room scenario with three independent speakers.](image)

The signal arriving at the microphone array is composed of an additive mixture of $C$ independent sound sources $s_c(t)$. In time domain, the samples of all $M$ microphones at sampling time $t$ can be stacked into a single $M \times 1$ vector, i.e.

$$z(t) = \sum_{c=1}^{C} s_c(t),$$

where $z(t) = [z(t, 1), \ldots, z(t, M)]^T$. We use bold symbols for vectors, i.e. $z(t)$, and plain symbols for scalars, i.e. $z(t, m)$. The vector $s_c(t)$ represents the $c^{th}$ sound source at sample time $t$. Each sound source is composed of a monaural recording $s_c(t)$ convolved with a Room Impulse Response (RIR) denoted by $h_c(t)$, i.e.

$$s_c(t) = h_c(t) \ast s_c(t),$$

where $\ast$ denotes the convolution operator. The RIR models the acoustic path from a sound source to each microphone as a set of $M$ FIR filters, which includes all reverberations and reflections caused by the room acoustics [47]. Modern office rooms are made of laminate flooring and concrete walls, which have a low acoustic absorption coefficient. Consequently, the reverberation time $RT_60$ may be very large, which significantly affects the performance of speech separation and speech recognition algorithms [29], [30], [31].

To cope with this environment, we propose the BSSD network, which iteratively extracts an unknown number of speakers from a multi-channel input mixture $z(t)$. During each iteration, the Direction Of Arrival (DOA) of the loudest speech source is estimated by a localization module, which correlates the input mixture against a pre-defined set of DOA bases. The DOA is subtracted from a spatial speech presence probability map, so that the second-loudest source is extracted during the subsequent iteration. The DOA information is then fed into a Neural Network (NN), which extracts and dereverberates the corresponding speech source. The network also predicts a speaker embedding vector for each extracted speech source, which is used to assign the utterance to a speaker for block-online processing. This iterative process is repeated until no new speaker embedding is found. Figure 2 illustrates two iterations of the BSSD network, which consists of the following modules: DOA bases, localization, beamforming and dereverberation, and speaker identification. In the following chapters, we will introduce each module in detail.

III. DOA BASES

As each source in Figure 1 has a direct line of sight towards the microphone array, it is possible to assign a unique DOA to each individual source in the mixture. Even if there is a significant amount of reverberation, there will always be an anechoic component in the RIR (i.e. the earliest peak) that corresponds to the DOA [47]. We therefore define a set of $D$ unique DOA vectors on a unit sphere around the microphone array, where each impinging sound wave is modeled as plane wave, i.e.

$$V(d, k, m) = e^{-i2\pi f_k \tau_{d,m}},$$

where $f_k$ is the frequency for index $k$ and $\tau_{d,m}$ is the time delay from a point on the sphere to the $m^{th}$ microphone, i.e.
The neural network also assigns a speaker embedding vector $V$ according to Eq. (1). The amplitude of the DOA vector $H_g$ is then used to extract and dereverberate the corresponding speech source from the multi-channel input mixture $Z(t)$. The neural network also assigns a speaker embedding vector $e_i$ to each enhanced source $i$.

$$\tau_{d,m} = \frac{\sqrt{(x_m - x_d)^2 + (y_m - y_d)^2 + (z_m - z_d)^2}}{c},$$

where $c$ is the speed of sound. The cartesian coordinates of the $n^{th}$ microphone are denoted by $x_m, y_m, z_m$, and $x_d, y_d, z_d$ are the coordinates of the $d^{th}$ point on the sphere. We define these points to be equally distributed on the surface of the sphere using a fibonacci spiral \cite{48}, i.e.

$$\phi_d = g \cdot d,$$

$$\theta_d = \arcsin \frac{d}{D - 1},$$

$$x_d = \cos \theta_d \cos \phi_d,$$

$$y_d = \cos \theta_d \sin \phi_d,$$

$$z_d = \sin \theta_d,$$

where $g = \pi(3 - \sqrt{5})$ is known as the golden angle \cite{48}, and $d = 1 \ldots D$ is the DOA index. We use a circular microphone array with $M$ channels. Hence, the array is flat and we cannot distinguish between positive and negative $z$ coordinates. It is therefore sufficient to only use half a sphere for the DOA bases. To assign a DOA index $d$ to a given RIR $h(t)$, we utilize GCC-PHAT \cite{49}, i.e.

$$\hat{d} = \arg\max_d \sum_{k=1}^K \left| H^H(k) \cdot V(d, k) \right|^2,$$

where $H$ represents the FFT of the RIR $h(t)$. Note that the amplitude of the DOA vector $V(d, k)$ is defined as 1 in Eq. \cite{41}.

### IV. Source Localization

To estimate the direction of a speech source relative to the microphone array, we again use GCC-PHAT to obtain a spatial speech presence probability map for the input mixture $z(t)$ and the DOAs $V(d, k)$. First, we transform the input mixture to the frequency domain using the Short-Time Fourier Transform (STFT), i.e.

$$z(t) \rightarrow Z(l, k),$$

where $Z(l, k)$ contains $M$ samples of frequency bin $k$ and STFT frame index $l$. Next, we compute the spatial speech presence probability map $\gamma \in [0 \ldots 1]$ as:

$$\gamma(l, k, d) = \frac{|Z^H(l, k) \cdot V(d, k)|^2}{|Z(l, k)|^2_2},$$

where $Z^H(l, k)$ is the Hermitian transpose of $Z(l, k)$.

#### A. Spatial Whitening

To separate speakers based on their location, Eq. \cite{41} utilizes the IPDs, which are encoded in the phase of the complex-valued input mixture $Z(l, k)$. However, it is well known that microphone array recordings are strongly correlated towards low frequencies \cite{49, 50, 51}. This is due to the fact that the wavelength of low frequencies is large compared to the aperture of the microphone array. As a consequence, the IPDs will be small, and the overall separation performance is degraded. To mitigate this effect, we decorrelate the noisy inputs $Z(l, k)$ using Zero-phase Component Analysis (ZCA) whitening \cite{52} from our previous works \cite{10, 7}. In particular, we use the whitening matrix

$$U(k) = E^H_T(k)D^{-\frac{1}{2}}_F(k)E^H_T(k),$$

where $E_T$ and $D_F$ are $M \times M$ sized eigenvector and eigenvector matrices of the real-valued spatial coherence matrix of the ideal isotropic sound field $\Gamma(k)$ \cite{47}. Its elements are given as $\Gamma_{i,j}(k) = \frac{\sin(2\pi f \cdot d_{i,j})}{2\pi f \cdot d_{i,j} c}$, and $d_{i,j}$ is the distance between the $i^{th}$ and the $j^{th}$ microphone. To avoid a division by zero, the diagonal elements of $D_F$ are loaded with a small constant $\epsilon = 10^{-3}$. We prefer ZCA whitening over PCA whitening, as the ZCA preserves the orientation of the distribution of the data \cite{52}. Using the whitening matrix $U(k)$, we rewrite Eq. \cite{41} as

$$\gamma_U(l, k, d) = \frac{|Z^H(l, k)U^H_T(k) \cdot U(k)V(d, k)|^2}{|U(k)Z(l, k)|^2_2 \cdot |U(k)V(d, k)|^2_2},$$

where $U(k)Z(l, k)$ can be recognized as the whitened input mixture, and $U(k)V(d, k)$ as whitened DOA vector. Figure \cite{3} demonstrates the effect of spatial whitening. Panel (a) shows $\gamma(l, k)$ for a single speaker and a matching DOA vector. Panel (b) shows $\gamma_U(l, k)$ with whitening. It can be seen that the separation performance is greatly increased for low frequencies.
maximum is denoted as $\hat{\gamma}$, network, which outputs an estimate of the desired signal $\gamma$ on the frame and frequency axes of speaker embeddings that map into $\gamma$-ance Distortionless Response BSSD architecture and the distance function in the following $z$ within the mixture this case, we stop the iterations and consider all speech sources $Z$ time-frequency bin of the input mixture due to the limited spatial resolution of the beamformer array.

energy, as multiple DOA indices may share the energy from the same speaker, gain mask on the output of the beamformer.

Well-established beamformers such as the Minimum Variance Distortionless Response (MVDR) beamformer [53] or the Generalized Eigenvalue (GEV) beamformer [54] use the signal statistics (i.e., the power spectral density matrices) to derive a set of beamforming weights $W(k) \in C$ in frequency domain. As those weights are static over time, the signal separation performance is limited especially in reverberant conditions [49]. Therefore, a beamformer is often used in conjunction with a post-filter [8]. The post-filter acts as a single-channel gain mask on the output of the beamformer.

1. Note that this algorithm is different to just sorting the DOA indices by energy, as multiple DOA indices may share the energy from the same speaker, due to the limited spatial resolution of the beamformer array.

Algorithm 1 Source localization

1: $P_Z(l,k) \leftarrow \frac{1}{M} \sum_{m=1}^{M} |Z(l,k,m)|^2$
2: $\gamma_W(l,k,d) \leftarrow \gamma_W(l,k) \cdot P_Z(l,k)$
3: $\gamma'_W(l,k,d) \leftarrow \gamma_W(l,k,d)$
4: $\mathcal{E} \leftarrow []$
5: $Y \leftarrow []$
6: while true do
7: $d \leftarrow \arg\max_{\hat{d}} \left( \sum_{l=1}^{L} \sum_{k=1}^{K} |\gamma_W(l,k,d)| \right)$
8: $(y(t), e) \leftarrow \text{BSSD}(z(t), d)$
9: $Y.append(g(t))$
10: if distance($\mathcal{E}, e$) $> \delta$ then
11: $\mathcal{E}.append(e)$
12: $\gamma'_W(l,k,:) \leftarrow \max(\gamma'_W(l,k,:) - \gamma_W(l,k,d), 0)$
13: else
14: break
15: end if
16: end while

In [10], we proposed the Complex-valued Neural Beamformer (CNBF), which combines the properties of a beamformer and a post-filter using a neural network. Unlike a statistical beamformer, the CNBF estimates a set of individual beamforming weights $W(l,k) \in C$ for each time-frequency bin. Those weights act as a spatio-temporal, complex-valued gain mask, which allows for a higher flexibility in the design of the beamformer, i.e., higher suppression rates or dereverberation. The CNBF uses complex-valued, non-holomorphic activation functions like vector normalization, phase normalization or conjugation. To back-propagate the complex-valued gradient, Wirtinger calculus is used [55], [56], [57]. A Tensorflow implementation of the CNBF network can be found at [6].

We extend the CNBF to include dereverberation and a speaker embedding vector. Figure 5 shows the architecture of the BSSD-FD network. The left branch performs beamforming and dereverberation, and the right branch outputs an embedding vector per utterance.

V. BSSD NETWORK - FREQUENCY DOMAIN

The STFT layer transforms the multi-channel input mixture to the frequency domain using Eq. (7). The STFT produces $L$ time frames and $K$ frequency bins per frame. Source separation is based on the DOA index $d$ from Algorithm 1, which is a scalar. The Adaption layer uses this input to modify the IPDs of the multi-channel input mixture in frequency domain, i.e.,

$$\tilde{Z}(l,k) = \left( U(k)V(d,k) \right)^* \odot \left( U(k)Z(l,k) \right), \quad (11)$$

where $^*$ denotes complex conjugation, and $\odot$ element-wise multiplication. The adaption layer performs two tasks: (i) It whitens the input signal $Z(l,k)$ as shown in Eq. (10). (ii) It subtracts the phase of the whitened DOA vector $V(d,k)$

2. https://github.com/mrbluke/CNBF
valued inputs $Z$ allows to scale, shift and mix the complex-valued linear layer \[10\]. We refer to Eq. (12) as weights in the tensor must be presented to the NN to train all complex-valued signal. Hence, during training, all possible DOA locations from which we want to extract the desired speech will have large IPDs. Hence, the NN by a small IPD, whereas all other signal components (i.e. interfering speakers) will have large IPDs. Hence, the NN from this direction (i.e. the desired signal) can be identified enabling it to distinguish between the desired and unwanted interfering speakers.

The left branch assigns an embedding vector to the enhanced output signal $y(t)$. The symbols next to each layer denote the dimensionality of the respective output tensor.

from the phase of the whitened input signal. This operation will align the phase of $\tilde{Z}(l,k)$ such that the IPDs for the direction of $V(\hat{d})$ are zero. Consequently, signals originating from this direction (i.e. the desired signal) can be identified by a small IPD, whereas all other signal components (i.e. interfering speakers) will have large IPDs. Hence, the NN sees the desired signal always at the same spatial location, enabling it to distinguish between the desired and unwanted signal components. Consequently, the NN extracts the speaker towards the direction of $V(\hat{d})$. We refer to Eq. (11) as the Analytic Adaption (AA). Hence, this system is abbreviated as BSSD-FD-AA.

Instead of modifying the phase of the input with the DOA vector, it is also possible to modify the input directly with a set of trainable weights, i.e.

$$\tilde{Z}(l,k) = A(\hat{d},k)Z(l,k), \quad (12)$$

where $A(\hat{d},k)$ is a complex-valued matrix of shape $M \times M$. It allows to scale, shift and mix the $M$ channels of the complex-valued inputs $Z(l,k)$ freely. Note that the DOA index $\hat{d}$ selects the location from which we want to extract the desired speech signal. Hence, during training, all possible $D$ DOA locations must be presented to the NN to train all complex-valued weights in the tensor $A$. Eq. (12) can be implemented as a complex-valued linear layer \[10\]. We refer to Eq. (12) as Statistic Adaption (SA). Hence, this system is abbreviated as BSSD-FD-SA.

### B. Beamforming and Dereverberation

The structure of the left branch of the NN in Figure 4 resembles a traditional filter-and-sum beamformer, which can be written as:

$$Y(l,k) = W^T(l,k)\tilde{Z}(l,k), \quad (13)$$

where $Y(l,k)$ denotes the beamformed output in frequency domain, and $W(l,k)$ are the beamforming filters. The inner vector product of Eq. (13) is computed before the inverse STFT layer in Figure 4. The NN estimates the weights $W(l,k)$ solely from the spatial information in $\tilde{Z}(l,k)$, which is obtained by the Norm layer. In particular, this layer normalizes the magnitude of the $M$ dimensional input vector $\tilde{Z}(l,k)$ to 1, and aligns its phase to the first microphone, i.e.

$$v_{\tilde{Z}}(l,k) = \frac{\tilde{Z}(l,k) \cdot \tilde{Z}^*(l,k, m = 1)}{\|\tilde{Z}(l,k) \cdot \tilde{Z}^*(l,k, m = 1)\|^2}. \quad (14)$$

Then, a bidirectional LSTM layer creates a latent space of $H$ neurons, followed by a dense layer with a complex-valued tahn activation function \[10\]. A linear layer outputs a set of unconstrained filter weights $W(l,k) \in \mathbb{C}$ to calculate the enhanced output $Y(l,k)$ as shown in Eq. (13). After the inverse STFT layer, we obtain the enhanced time-domain signal $y(t)$. By using a neural beamformer, the design goal is not limited to MVDR constraints or similar concepts \[10\]. In fact, we can also include a dereverberation objective by using an appropriate loss function for the NN. In particular, we use the negative SI-SDR \[58\] between the output $y(t)$, and a clean anechoic reference utterance $r(t)$, i.e.

$$\mathcal{L}_{\text{SI-SDR}} = -10 \log_{10} \left( \frac{\|\alpha r(t)\|^2}{\|\alpha r(t) - y(t)\|^2} \right), \quad (15)$$

where $\alpha = \frac{y(t)^T \tau(t)}{r(t)^T \tau(t)}$. We use $r(t) = s_c(t)$ from Eq. (2) as anechoic reference signal.

### C. Speaker Identification

The right branch of the NN in Figure 4 extracts an embedding vector $e$ to identify the speaker in the enhanced output signal $Y(l,k)$. The embedding vector maps the utterance into a feature space where distances correspond to speaker similarity \[38\]. Therefore, the NN must be agnostic to the spoken text, and only rely on the speaker characteristics embedded in the waveform. We use the log-power spectral density $\log(\|Y(l,k)\|^2)$ as input features. Then, a series of 6 convolutional layers with a filter length of 10, and increasing dilation factors of $(1,2,4,8,16,32)$ frames create a latent space of $L \times E$ dimensional embeddings. We use a softplus activation function and layer normalization after each convolutional layer. A skip connection is added between every two convolutional layers. The $L$ time frames are averaged to obtain a single, $E$ dimensional embedding for the whole

\[3\]The softplus activation function is defined as $f(z) = \log(1 + e^z)$. 

---

Fig. 4: Layers of the frequency-domain BSSD-FD network. The left branch performs beamforming and dereverberation, the right branch assigns an embedding vector to the enhanced output signal $y(t)$. The symbols next to each layer denote the dimensionality of the respective output tensor.
When the batch size $P$ utterances. More formally, this procedure can be written as:

$$
L_{\text{TL}} = \sum_{B=1} \left[ |e_a - e_p|_2 - |e_a - e_n|_2 + \beta \right]_+, \quad (16)
$$

where the embedding $e_a$ denotes an anchor, $e_p$ is an embedding from the same speaker as the anchor (positive example), and $e_n$ is an embedding from a different speaker (negative example). In a batch of $B$ utterances, there can be as many as $B^3$ triplets. It is therefore crucial to only select a subset of valid triplets, where the positive example is from the same speaker as the anchor, and the negative example belongs to a different speaker. Further, we only need to consider triplets where the loss $L_{\text{TL}}$ is actually greater than zero. To select relevant triplets, we utilize **Hard Triplet Mining** [59], where we select the hardest positive and negative example per anchor. In particular, we randomly select $P$ utterances from $B$ speakers, where we determine the largest distance $|e_a - e_p|_2$ between an anchor and a positive example within the $P$ utterances per speaker, and the smallest distance $|e_a - e_n|_2$ between an anchor and a negative example from the $P(B-1)$ remaining utterances. More formally, this procedure can be written as:

$$
L_{\text{TL-HTM}} = \frac{1}{B \cdot P} \sum_{i=1}^{B} \sum_{j=1}^{P} \left[ \beta + \max_{p=1\ldots P} \left( |e_a - e_p|_2 \right) \right. \\
- \min_{n \neq i; j} \left( |e_a - e_n|_2 \right). \quad (17)
$$

When the batch size $P \cdot B$ is small, the embeddings may collapse into a single point during training [60]. To avoid this, we propose to minimize the cross-entropy between embeddings of different speakers as follows:

$$
L_{\text{TL-CE}} = -\frac{1}{(B^2 - B)P^2} \sum_{a=1}^{B} \sum_{n \neq a}^{B} \sum_{i=1}^{P} \sum_{j=1}^{P} \log \left( \left( \frac{e_a - e_i}{|e_a - e_i|_2} \right) \cdot \left( \frac{e_a - e_j}{|e_a - e_j|_2} \right) \right), \quad (18)
$$

where $\hat{e} = \frac{e}{|e|_2}$ is the magnitude-normalized embedding vector $e$. This regularization ensures that the embeddings $e_a$ and $e_n$ will be different. The overall cost function for the entire BSSD-FD architecture is then defined as:

$$
L_{\text{BSSD-FD}} = L_{\text{SL-SDR}} + \lambda_1 L_{\text{TL-HTM}} + \lambda_2 L_{\text{TL-CE}}, \quad (19)
$$

where $\lambda_1$ and $\lambda_2$ are weights for the individual terms.

### D. Distance Measure

In order to determine whether two embeddings $e_1$ and $e_2$ belong to the same speaker, we use the euclidean distance $|e_1 - e_2|_2$. If the distance falls below a certain threshold $\delta$, we consider the two embeddings to belong to the same speaker. If it exceeds the threshold, the speakers are considered different. Hence, two types of errors exist: (i) A false positive is triggered when two embeddings from two different speakers are incorrectly classified as belonging to the same speaker, which we measure using the False Acceptance Rate (FAR), i.e.

$$
\text{FAR}(\delta) = \frac{1}{(B^2 - B)P^2} \sum_{a=1}^{B} \sum_{n \neq a}^{B} \sum_{i=1}^{P} \sum_{j=1}^{P} \mathbb{1} \left( |e_a - e_i|_2 < \delta \right), \quad (20)
$$

where $\mathbb{1}(x)$ denotes an indicator function, i.e.

$$
\mathbb{1}(x) = \begin{cases} 1, & \text{if condition } x \text{ is true.} \\ 0, & \text{otherwise.} \end{cases} \quad (21)
$$

(ii) A false negative is triggered when two embeddings from the same speaker are classified as belonging to different speakers, which we measure using the False Rejection Rate (FRR), i.e.

$$
\text{FRR}(\delta) = \frac{1}{B(P^2 - P)} \sum_{a=1}^{B} \sum_{j=1}^{P} \sum_{j \neq i}^{P} \mathbb{1} \left( |e_a - e_j|_2 > \delta \right). \quad (22)
$$

The FAR is positively correlated to the decision threshold $\delta$, and the FRR is correlated negatively. The value at which the FAR and FRR are equal, is known as the Equal Error Rate (EER). It is determined by:

$$
\hat{\delta} = \arg \min_{\delta} |\text{FAR}(\delta) - \text{FRR}(\delta)| \quad \text{EER} = \text{FAR}(\hat{\delta}) = \text{FRR}(\hat{\delta}), \quad (23)
$$

where $\hat{\delta}$ is considered the optimal threshold belonging to the EER.

### VI. BSSD NETWORK - TIME DOMAIN

With the recent success of time-domain speech separation algorithms [15], [16], [17], [25], we also formulate a time-domain variant of our BSSD network. The beamforming and dereverberation operations can be formulated analogously to the frequency domain, but the NN will train faster when using real-valued data. Figure [5] shows the architecture of the BSSD-TD network. Similar to the frequency-domain variant, the left branch performs beamforming and dereverberation, and the right branch outputs an embedding vector per utterance.
Fig. 5: Layers of the time-domain BSSD-TD network. The left branch performs beamforming and dereverberation, the right branch assigns an embedding vector to the enhanced output signal $y(t)$.

A. Speaker Separation

Analogous to the frequency-domain network, source separation is based on the Adaptation layer, which uses the DOA index $\hat{d}$ from Algorithm 1 to modify the ITD of the input signal $z(t)$. By rearranging Eq. (11), we can formulate an identical operation in time-domain, i.e.

$$
\tilde{Z}(l, k, m) = (U^T(k, m)V(\hat{d}, k))^* \cdot (U^T(k, m)Z(l, k)),
$$

$$
= \sum_{i=1}^{M} U^H(k, m)V^*(\hat{d}, k)U(k, m, i) \cdot Z(l, k, i),
$$

$$
= \sum_{i=1}^{M} V'(\hat{d}, k, m, i) \cdot Z(l, k, i),
$$

(24)

where we can identify the convolutional kernel $V'(\hat{d}, k, m, i)$ in frequency domain. We can see from Eq. (3), that the DOA $V'(\hat{d}, k)$ resembles $M$ sinc pulses with a positive time-delay $\tau_{d,m}$, and Eq. (9) shows that the elements of the whitening matrix $U(k, m, i)$ are real-valued. Therefore, $V'(\hat{d}, k, m, i)$ will be a causal IIR filter in time domain [61], which we truncate to $T_A$ samples to obtain the FIR filter $v'(\hat{d}, t_A, m, i)$ by using the inverse FFT. This allows to formulate the time-domain adaption layer as:

$$
\tilde{z}(t, m) = \sum_{i=1}^{M} z(t, i) \circ v'(\hat{d}, t_A, m, i),
$$

(25)

which can be implemented using a single convolution layer. Similar to the frequency-domain adaption layer, Eq. (25) synchronizes the ITD to be zero for signals originating from the direction of $V'(\hat{d})$, i.e. the desired signal. The subsequent NN sees the desired signal always at the same spatial location, which makes it easier to distinguish between the desired and unwanted signal components. Consequently, the NN extracts the speaker towards the direction of $V'(\hat{d})$. We refer to Eq. (25) as Analytic Adaption (AA). Hence, this system is abbreviated as BSSD-TD-AA.

Instead of modifying the ITDs of the input signal with the DOA vector, it is also possible to replace the fixed convolutional kernels $v'(\hat{d}, t_A, m, i)$ with a set of trainable weights, i.e.

$$
\tilde{z}(t, m) = \sum_{i=1}^{M} z(t, i) \otimes a(\hat{d}, t_A, m, i),
$$

(26)

where $a$ is a tensor of shape $(D, T_A, M, M)$, and $T_A$ is the filter length of the learnable convolution kernels. This allows to scale, shift and mix the $M$ channels of the input signal $z(t)$ freely. Note that the DOA index $\hat{d}$ provides the location from which we want to extract the desired speech signal. Hence, during training, all $D$ possible DOA locations must be presented to the NN to train the weights $a$. Note that Eq. (26) can be implemented as a convolution layer. We refer to Eq. (26) as Statistic Adaption (SA). Hence, this system is abbreviated as BSSD-TD-SA.

B. Beamforming and Dereverberation

The structure of the left branch of the NN in Figure 5 resembles a time-domain beamformer [49], where the first convolution layer right after the adaption layer transforms the time-domain input $\hat{z}(t)$ into a latent space $z'(l, h)$ with $L$ frames and $H$ filters. The stride of this convolution layer is set to $\frac{H}{2}$, and the activation function is linear.

Similar to the frequency-domain beamformer, filtering is performed in latent space. The beamforming weights $w'(l, h)$ are estimated from the spatial information embedded in $z'(l, h)$, using layer normalization, followed by a bidirectional LSTM layer, a dense layer with tanh activation, and a linear layer. The linear layer allows the output to freely choose the amplitude and phase of the beamforming weights. The enhanced output $y'(l, h)$ is obtained by

$$
y'(l, h) = w'(l, h) \odot z'(l, h),
$$

(27)

where all variables are of shape $L \times H$. Finally, a deconvolution layer with a linear activation function produces the enhanced time-domain signal $y(t)$. Analogous to the BSSD-FD architecture, we use the negative SI-SDR from Eq. (15) between the output $y(t)$, and a clean anechoic reference utterance $r(t)$.

C. Speaker Identification

The right branch of the NN in Figure 5 extracts an embedding vector $e$ to identify the speaker in the enhanced output signal $y(t)$. The NN is identical to the BSSD-FD architecture,
except for the input layer which uses the enhanced signal \( y'(l, h) \) as input features. Identically to Eq. (19), the overall cost function for the entire BSSD-TD architecture is defined analogously to the BSSD-FD architecture given in Eq. (19).

VII. BLOCK ONLINE PROCESSING

For real-time applications, it is possible to use the BSSD system in block-online mode. We split the input mixture \( z(t) \) into blocks of equal length. Each block \( b \) is iteratively processed using Algorithm 1. It returns the DOA index \( d \), a list \( \hat{Y}_b \) of extracted speakers \( y(t) \), and a list \( \hat{E}_b \) of speaker embeddings \( e \). Figure 6 illustrates the block-online processing scheme of the BSSD system for \( C = 2 \) speakers and 4 blocks. Note that the speakers may change their position from block to block, i.e., they may walk around, switch places, etc. Therefore, both the DOA and speaker embedding are extracted for each source in each block. The process of assigning each extracted source block to the same speaker identity is referred to as diarization [36], [37], by using Algorithm 2.

Algorithm 2 Diarization in block-online mode.

```plaintext
1: \( \hat{Y} \leftarrow \emptyset \)
2: \( \hat{E} \leftarrow \emptyset \)
3: for all blocks \( b \) do
4:   for \( c = 1 \) : length(\( \hat{E}_b \)) do
5:     if \( \min(\|\hat{E} - \hat{E}_b(c)\|_2) > \delta \) then
6:       \( \hat{E} . \text{append}(\hat{E}_b(c)) \)
7:     else
8:       \( i \leftarrow \text{argmin}(\|\hat{E} - \hat{E}_b(c)\|_2) \)
9:       \( \hat{Y}(i) . \text{append}(\hat{Y}_b(c)) \)
10:   end if
11: end for
12: end for
```

First, we initialize empty lists for all speakers \( \hat{Y} \) and all embeddings \( \hat{E} \). Then, we iterate over all blocks \( b \), where Algorithm 1 is executed for each block. It returns a list \( \hat{Y}_b \) of extracted speakers and a list \( \hat{E}_b \) of speaker embeddings for that block. Next, we iterate over each extracted source \( c \) within that block, and we compare the distance of the embedding \( \hat{E}_b(c) \) against all embeddings \( \hat{E} \). If the threshold \( \delta \) (see Eq. 23) is exceeded, we have found a new speaker. In that case, this speaker is added to the list of known embeddings \( \hat{E} \). Otherwise, we have found an utterance belonging to a known embedding. In that case, we determine the index \( i \) of that embedding, and append the source \( \hat{Y}_b(c) \) to the speaker at position \( \hat{Y}(i) \). To preserve the time alignment of each extracted speaker, we append a block of silence to each source in \( \hat{Y} \) that did not receive an update. This may happen if a speaker is silent during block \( b \).

A trade-off has to be made when choosing the block length \( T_B \). If it is too large, short utterances followed by periods of silence might not get detected. If it is too small, the predicted embeddings may be inaccurate, causing Algorithm 2 to assign the sources \( \hat{Y}_b(c) \) to the wrong speaker. We examine this behavior in our experiments.

VIII. RIR RECORDINGS

We use both recorded and simulated RIRs to generate spatialized recordings with Eq. (4). Real RIRs are obtained through multi-channel room impulse response measurements, and simulated RIRs are obtained by the Image Source Method (ISM) [23], [62].

A. Real RIRs

To obtain realistic RIRs, we use a circular microphone array with \( M = 6 \) channels and a diameter of 92.6mm [63], and a 5W measurement loudspeaker. To drive the loudspeaker from a Linux-based PC with ALSA [64], we use the PlayRec Python module [65], which simultaneously plays and records audio from a sound card. We use an exponential chirp with a duration of 5s sweeping from 24kHz down to 20Hz as excitation signal [47]. However, we only use a bandwidth of 8kHz for our experiments. We recorded 120 6-channel RIRs in 24 different, fully furnished office rooms with a reverberation time \( RT_{60} \in [200 \ldots 900] \) ms. The distance from the loudspeaker to the microphone array was varied from 1m...3m, and the direction was chosen randomly. Figure 7 shows the recording setup. We augmented the number of RIR recordings to 720 by virtually rotating the array by \( 6 \times 60^\circ \), i.e., shifting the microphone channels.

Fig. 6: Block-online processing mode of the BSSD system, showing a mixture of \( C = 2 \) speakers being split into 4 blocks. Each block is processed separately using Algorithm 1.

Fig. 7: RIR recording setup using a 5W measurement loudspeaker and a 6-channel microphone array [63].
B. Simulated RIRs

To obtain simulated RIRs, we further generated 720 artificial RIRs for the same array geometry with 6 channels, but with a shorter reverberation time $RT_{60} \in [200 \ldots 400]$ms, which is randomly chosen. The room is modeled as a simple rectangular shoebox with random dimensions ranging from 3m $\ldots$ 6m, where the microphone array and the sound sources are placed randomly. RIR generation was done using the Image Source Pyroomacoustics [66].

IX. EXPERIMENTS

A. Experimental Setup

1) Speech mixtures: We use the WSJ0 speech database which contains 12776 utterances from 101 different speakers for training, and 5895 utterances from 18 different speakers for testing. To generate mixtures, we use the wsj0-2mix from [12], which we extended to 3 and 4 speakers. To generate reverberated, multi-channel mixtures from Eq. (1), we convolve the monaural signals with both the real and simulated RIRs, as described in Section VIII. All recordings use a sample rate of $f_s = 16kH$.

2) DOA bases: We use $D = 100$ DOA bases, which are equally distributed on a sphere, as shown in Figure 8. This provides an average spatial resolution of about $13^\circ$, which equates to two persons standing right next to each other at a distance of 1m. Note that the number of DOA vectors $D$ can be changed without re-training the NN. Clearly, we want to use a different DOA index $d$ for each source $s_c$, which contains 12776 utterances from 101 different speakers for training, and 5895 utterances from 18 different speakers for testing.

3) BSSD-FD system: For the BSSD-FD network in Figure 4, we use a FFT length of 1024 samples, and an overlap of 75%. This results in $K = 513$ frequency bins. Further, we have $M = 6$ microphones as determined by the RIR recordings. The beamforming branch uses $H = 500$ neurons to create the beamforming weights $W(l,k)$, and to predict the enhanced signal $Y(l,k)$. The identification branch uses an embedding dimension of $E = 100$ to predict the speaker embeddings $e$.

4) BSSD-TD system: For the BSSD-TD network in Figure 5 we use a filter length of $T_A = 100$ samples for the filter kernels in the adaption layer in Eq. (25) and (26). The first convolutional layer uses a filter length of 200 samples and a stride of 50 samples to create a latent space of $H = 500$ neurons. The beamforming branch predicts the beamforming weights $u_f(l)$ and the enhanced signal $y_f(l)$ in latent space. This signal is transformed back to time domain using the deconvolution layer, which uses a filter length of 200 samples, a stride of 50 samples, and overlap-add to produce the enhanced signal $y(l)$. The identification branch uses an embedding dimension of $E = 100$ to predict the speaker embeddings $e$.

B. Related Systems

To compare our BSSD system against other state-of-the-art speech separation algorithms, we evaluate Conv-TasNet [17] and PIT with spatial features [23]. To account for dereverberation, we apply the WPE algorithm from [35] prior to Conv-TasNet.

1) Conv-TasNet: Conv-TasNet separates 2 speakers in time domain. It operates on chunks of 4s of audio, where it separates the two speakers in a latent space by using a speech mask in $[0 \ldots 1]$. The mask is obtained from a series of convolutions. The system operates on single-channel inputs, therefore we only use one output of the WPE algorithm. We use the implementation of [17].

2) Spatial PIT: Spatial PIT separates 2 speakers in frequency domain. It uses log-specograms and the sine and cosine of the IPDs of frequency-domain, multi-channel mixtures to predict a speech mask for each speaker [13], [23]. This mask is used to construct a frequency-domain beamformer [7]. Note that there is no explicit dereverberation constraint, but the target speech mask is obtained from the anechoic reference signal $r(t)$. Hence, the beamformer will remove late echoes.

C. Training

All four variants of the BSSD network are trained on mixtures of $C = 2$ sources, where each mixture $z(t)$ is truncated to 5s length. The location (i.e. the DOA index $d$) of each source is chosen randomly for each example. We use a batch size of 60 mixtures from the 101 speakers of the WSJ0 training set. To enable efficient triplet mining with Eq. (17), we use $P = 3$ different utterances from $B = 20$ speakers for the first source $s_{c=1}(t)$ of each mixture. The second source $s_{c=2}(t)$ is chosen randomly from the remaining 100 speakers from the WSJ0 training set. We use the clean first source as reference utterance, i.e. $r(t) = s_{c=1}(t)$. The ground truth DOA index $\hat{d}$ is used to train the network. We use $\lambda_1 = 10^{-2}$ and $\lambda_2 = 10^{-4}$ for the cost function in Eq. (19). This ensures that a batch of 60 mixtures from the 101 speakers of the WSJ0 training set. We use the clean first source as reference utterance, i.e. $r(t) = s_{c=1}(t)$. The ground truth DOA index $\hat{d}$ is used to train the network. We use $\lambda_1 = 10^{-2}$ and $\lambda_2 = 10^{-4}$ for the cost function in Eq. (19). This ensures that the beamforming path is trained faster than the identification path, as the latter depends on the former. As the combination of the different RIRs and WSJ0 utterances allows for billions of combinations, we randomly create new batches for training and validation for each epoch. Adam is used as optimizer [67], with a learning rate of $10^{-3}$. A Tensorflow implementation of the BSSD network can be found at [4].

D. Testing

We compare the frequency-domain (FD) and time-domain (TD) variants of our BSSD system, as well as the analytic adaption (AA) and statistic adaption (SA) layers introduced in Section VII and VII. Further, we use the Conv-TasNet and spatial PIT as baseline systems. We test the BSSD network both in offline mode in block-online mode.

1) Offline Mode: In offline mode, we use 5s long mixtures of $C = \{1, 2, 3, 4\}$ speakers from the test set. To be able to test the performance of the speaker separation and speaker identification modules separately, we use the ground truth DOA index $\hat{d}$ as input to the BSSD network. We report the separation and dereverberation performance in terms of SI-SDR using Eq. (15). Further, we report the WER using the

*https://github.com/rrbluke/BSSD
Google Speech-to-Text API \cite{GoogleASR} to perform ASR. Speaker identification performance is reported in terms of EER on the enhanced output, using Eq. (23).

2) Block-online Mode: In block-online mode, we use 20s long mixtures from the test set, which we divide into \( N_b \) blocks of \( T_B = \{1, 2.5, 5\} \) s length. Each block is processed by Algorithm 1 which outputs the DOA index \( \hat{d} \), a list of extracted signals \( y_b \), and a list of speaker embeddings \( E_{b,c} \) for each block \( b \). Then, Algorithm 2 is used to assign the extracted utterances of each block to the same speaker. This solves the speaker permutation problem. We report the SI-SDR, WER and the Block Error Rate (BER) for the extracted speakers. The BER indicates the percentage of falsely assigned blocks due to erroneous embeddings. It is determined by comparing the speaker embedding of the reference utterance \( r \) due to erroneous embeddings. It is determined by comparing the speaker embedding of the reference utterance \( r \) against the extracted chunks \( y_{b,c} \) for each speaker \( c \), i.e.

\[
\text{BER} = \frac{1}{C \cdot N_b} \sum_{c=1}^{C} \sum_{b=1}^{N_b} \mathbf{1}\left(|e_{r,c} - e_{b,c}|_2 > \delta\right)
\]  

(28)

X. RESULTS

A. Source Localization

First, we report the performance of the localization stage, i.e. Algorithm 1 using a mixture with \( C = 3 \) speakers and \textit{real} RIRs. Figure 8 illustrates the \( D = 100 \) DOA positions, arranged on a unit sphere according to Eq. (5). The color gradient indicates the spatial speech presence probability map \( \gamma^W(d) \), which is obtained by summing Eq. (10) over all frequencies \( K \) and time frames \( L \). The markers \( X_1, X_2 \) and \( X_3 \) indicate the positions of the three speakers. Panel (a) illustrates \( \gamma^W(d) \) for the first iteration of Algorithm 1, panel (b) for the second iteration, panel (c) for the third iteration, and panel (d) for the fourth iteration. All three sources are localized during the first three iterations, while the fourth iteration only sees a faint reflection of one of the three sources. Therefore, the speaker embedding for this reflection will already be known, which serves as the stopping criterion of Algorithm 2.

B. Separation Performance

Figure 9 shows the performance of the BSSD-TD-SA model with the same three speakers as in Figure 8. From panel (a) it can be seen that there is a significant amount of reverberation in the input mixture \( z(t) \). Panel (b) and (d) show the extracted and dereverberated signals of two male speakers. Panel (c) shows the extracted and dereverberated signal of a female speaker. Even though speaker 2 and 3 are located very close to each other, they can still be separated. Note that the spatial resolution is only determined by the number of DOA vectors \( D \), and the BSSD model is trained independently of the localization stage in Algorithm 1.

C. Speaker Identification

Figure 10 illustrates a 2D projection of the embeddings for each of the 101 WSJ0 speaker identities, which were obtained using BSSD-TD-SA model. For each speaker, 20 random utterances have been used, which have been reverberated and spatialized using both the \textit{real} and \textit{simulated} RIRs. It can be seen that each speaker identity can be distinguished.

D. Offline mode

Table I reports SI-SDR, WER and EER for the \textit{real} RIRs. For \( C = 1 \) speaker, the BSSD models only perform dereverberation. Hence, the SI-SDR is the highest for this case. The low WER of 9.65\% indicates that Google-ASR recognizes reverberant audio quite well. However, all BSSD models could lower the WER even further. Also, the EER is lowest for one speaker. This is to be expected, as no interfering components of other speakers reduce the quality of the speaker embeddings \( e \). For \( C = 2 \) speakers, it can be seen that all BSSD models outperform Conv-TasNet and spatial PIT, even though Conv-TasNet is aided by WPE, and spatial PIT has explicitly been trained to perform dereverberation as well. Conv-TasNet could only achieve a WER of 48.71\%. Spatial PIT only achieves a WER of 42.27\%. It uses a static beamformer, which performs poorly in reverberant environments.\footnote{Figure 8: Unit sphere with \( D = 100 \) equidistant DOA points and a circular microphone array with \( M = 6 \) channels. Panel (a) - (d) show the spatial speech presence probability map \( \gamma^W \) during consecutive iterations of Algorithm 1.}

For \( C = 3 \) and 4 speakers, performance of the BSSD architecture drops, as expected. I.e.: the SI-SDR gets lower, and both the WER and EER rise. The statistic adaption (SA) outperforms the analytic adaption (AA) variants for all number of speakers. This is also expected, as the SA variant allows the network to find an optimal transformation to separate the speakers both in time- and frequency domain, while the AA enforces a fixed scheme for spatial whitening and source localization. When comparing the time-domain (TD) to the frequency-domain (FD) variants, it can be seen that the FD models perform slightly better in terms of EER. This indicates that the speaker embeddings are easier to estimate in frequency
Fig. 9: Performance plot of the BSSD-TD-SA model with $C = 3$ speakers and real RIRs. (a) STFT plot of the first microphone of the input mixture $z(t)$. The $T_{60}$ of the reverb is approximately 650ms. (b) STFT plot of the clean first source signal $r_1(t)$. (c-e) STFT plots of the extracted and dereverberated sources $y_i(t)$.

Table I reports SI-SDR, WER and EER for the significantly shorter simulated RIRs. Consequently, all systems perform better in all scores. In these almost ideal conditions, Google-ASR achieved a WER of 3.04% for a single speaker without any enhancement. However, all BSSD variants also achieved a lower WER as well. Also, Conv-TasNet and spatial PIT perform better compared to the real RIRs. However, Conv-TasNet still could not separate the speakers perfectly. The static beamformer of spatial PIT performs quite well for the short simulated RIRs, achieving a WER of 17.1%. However, all BSSD variants achieved a lower WER for $C = 2$ speakers. Again, the FD variants perform slightly better than the TD models, and the SA layer outperforms the AA layer.

Fig. 10: t-SNE plot of the extracted speaker embeddings, using the 101 identities of the WSJ0 corpus.

Table II reports SI-SDR, WER and EER for the simulated RIRs in offline mode.

Table III reports SI-SDR, WER and BER for the real RIRs, for block lengths of $T_B = 1s, 2.5s$ and $5s$. We only performed

E. Block-online mode

domain, as they are calculated from the enhanced spectrograms (see Figure 4).
these experiments on the SA variants of the BSSD network, as the SA layer consistently outperforms the AA layer. The SI-SDR and WER are worse compared to offline mode, as many sources of errors cumulate throughout the processing chain. I.e.: Algorithm [1] may produce wrong DOA indices for short blocks and many speakers. Consequently, speaker separation is poor, resulting in erroneous speaker embeddings. Further, both the speaker separation and speaker identification modules introduce errors on their own. For the shortest block length of $T_B = 1s$, there are 20 blocks for 20s of audio. In order to achieve a perfect BER score, the embeddings for the same speaker in all 20 blocks must be identical (see Eq. (28)). If the speaker is silent in one or more blocks, a perfect BER score cannot be achieved. Clearly, performance is better for larger block lengths and fewer speakers. For $C = 2$ speakers and a block length of $T_B = 5s$, the WER is 34.79% for the FD variant, and 28.51% for the TD variant. The BER is 10% for the FD variant, and 4.5% for the TD variant. In contrast to the experiments in offline mode, all scores are slightly better for the TD models.

TABLE III: Speech separation and dereverberation performance for the real RIRs in block-online mode.

| model         | $T_B$ | SI-SDR  | WER   | BER   |
|---------------|-------|---------|-------|-------|
| BSSD-FD-SA    | 1.0s  | 3.80 dB | 66.94 %| 37.40 %|
|               | 2.5s  | 7.05 dB | 44.22 %| 15.88 %|
|               | 5.0s  | 8.64 dB | 34.79 %| 10.00 %|
| BSSD-TD-SA    | 1.0s  | 3.04 dB | 75.68 %| 48.90 %|
|               | 2.5s  | 5.19 dB | 59.94 %| 27.13 %|
|               | 5.0s  | 6.73 dB | 52.98 %| 14.75 %|
|               | 4.0s  | 2.49 dB | 78.21 %| 64.40 %|
|               | 2.5s  | 3.71 dB | 71.96 %| 43.75 %|
|               | 5.0s  | 4.76 dB | 68.07 %| 35.00 %|
|               | 1.0s  | 4.49 dB | 60.67 %| 25.30 %|
|               | 2.5s  | 7.04 dB | 36.24 %| 10.75 %|
|               | 5.0s  | 8.47 dB | 28.51 %| 4.50 %|
|               | 3.0s  | 3.30 dB | 74.11 %| 39.70 %|
|               | 2.5s  | 4.81 dB | 63.82 %| 23.88 %|
|               | 5.0s  | 6.31 dB | 48.68 %| 22.50 %|
|               | 4.0s  | 2.70 dB | 76.32 %| 47.85 %|
|               | 2.5s  | 3.93 dB | 70.83 %| 32.25 %|
|               | 5.0s  | 4.86 dB | 65.14 %| 32.50 %|

Table IV reports SI-SDR, WER and BER for the simulated RIRs, for block lengths of $T_B = 1s$, $2.5s$ and $5s$. All scores are better compared to the significantly longer real RIRs. For $C = 2$ speakers and a block length of $T_B = 5s$, the WER is 19.80% for the FD variant, and 16.75% for the TD variant. The BER is 3.25% for the FD variant, and 1.25% for the TD variant. Again, In contrast to the experiments in offline mode, all scores are slightly better for the TD models.

F. Model complexity

Table V reports the number of trainable parameters per variant of the BSSD network. The frequency domain (FD) variants use mostly complex-valued weights, which are counted as 2 real-valued weights. Hence, these models are significantly larger than the time domain (TD) variants. The size of the statistic adaption (SA) layer in the time domain network is comparatively small with 720,000 parameters. However, the analytic adaption (AA) layer requires additional convolutions from Eq. (25). Similar to Conv-TasNet, the time domain variant also has the advantage of a small step size of 50 samples. The number of parameters for speaker identification is almost the same for all variants.

TABLE IV: Speech separation and dereverberation performance for the simulated RIRs in block-online mode.

| model         | $T_B$ | SI-SDR  | WER   | BER   |
|---------------|-------|---------|-------|-------|
| BSSD-FD-SA    | 1.0s  | 4.24 dB | 58.08 %| 27.55 %|
|               | 2.5s  | 8.13 dB | 27.70 %| 7.88 %|
|               | 5.0s  | 10.67 dB| 19.80 %| 3.25 %|
| BSSD-TD-SA    | 1.0s  | 3.49 dB | 75.03 %| 40.88 %|
|               | 2.5s  | 6.46 dB | 46.98 %| 17.50 %|
|               | 5.0s  | 7.91 dB | 35.83 %| 11.75 %|
|               | 2.5s  | 2.89 dB | 81.74 %| 49.40 %|
|               | 5.0s  | 5.25 dB | 52.42 %| 24.63 %|
|               | 6.0s  | 6.50 dB | 49.52 %| 23.75 %|
|               | 1.0s  | 5.82 dB | 51.17 %| 18.55 %|
|               | 2.5s  | 10.94 dB| 18.21 %| 3.63 %|
|               | 5.0s  | 11.91 dB| 16.75 %| 1.25 %|
|               | 1.0s  | 4.40 dB | 73.33 %| 30.50 %|
|               | 2.5s  | 7.75 dB | 47.37 %| 15.25 %|
|               | 5.0s  | 9.66 dB | 39.12 %| 9.75 %|
|               | 1.0s  | 3.10 dB | 81.24 %| 37.65 %|
|               | 2.5s  | 5.11 dB | 60.12 %| 26.00 %|
|               | 5.0s  | 7.80 dB | 52.99 %| 10.75 %|

XI. CONCLUSION

In this paper, we introduced the Blind Speech Separation and Dereverberation (BSSD) network, which performs simultaneous speaker separation, dereverberation and speaker identification in a single neural network. We proposed four variants of our system, which operate in frequency-domain and time-domain, and use analytic adaption and statistic adaption layers to perform blind speaker separation. We have shown that 100 DOA bases provide enough spatial resolution to separate up to four speakers. Further, we proposed the block-online mode to process longer audio recordings, as they occur in meeting scenarios. In our experiments, we could show that the BSSD network outperforms similar state-of-the-art algorithms for speaker separation in terms of SI-SDR and WER.

REFERENCES

[1] D. Wang and J. Chen, “Supervised speech separation based on deep learning: An overview,” IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 26, no. 10, pp. 1702–1726, 2018.
[2] J. Barker, R. Marxer, E. Vincent, and S. Watanabe, “The fourth CHiME speech separation and recognition challenge: Dataset, task and base- lines,” in IEEE 2015 Automatic Speech Recognition and Understanding Workshop (ASRU), 2015.
[3] E. Vincent, S. Watanabe, J. Barker, and R. Marxer, “The fourth CHiME speech separation and recognition challenge: Dataset, task and base- lines,” in Proc. of the 4th Intl. Workshop on Speech Processing in Everyday Environments (CHiME 2016), 2016.
