Speech Separation in Wireless Acoustic Sensor Networks

November 23, 2022

Mads Græsbøll Christensen

Audio Analysis Lab, CREATE
Aalborg University, Denmark
Website: http://audio.create.aau.dk
Youtube: http://tinyurl.com/yd8mo55z

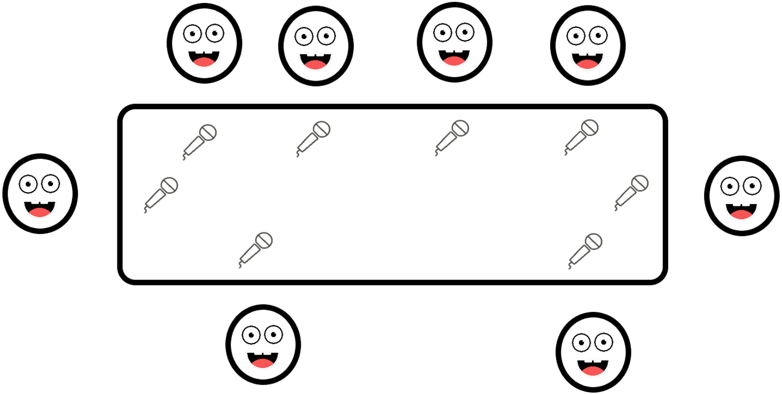
AALBORG UNIVERSITY
DENMARK

Motivation

Example: conventional microphone array

Motivation

What if . . . ?



Motivation

Conventional array

- array geometry known
- synchronized sampling
- centralized processing

Wireless acoustic sensor network (WASN)

- array geometry unknown
- unsynchronized sampling
- centralized/distributed processing

Some applications

- Meetings
- Hearing aids
- Hearables
- Smart homes
- Digital assistants
- Surveillance
- IoT

Outline

Multi-channel Wiener Filtering

The speech enhancement problem

From K noisy microphone signals $\{y_k(n)\}_{k=1}^{K}$, we wish to estimate a target speech component $s(n)$ to improve the

- speech intelligibility and quality
- speech recognition performance

However,

- we have more unknowns than observations so
- we need prior information about the speech, room, and/or noise to solve the problem!
- even defining the target speech is difficult in WASN!

Multi-channel Wiener Filtering
Mathematical model

At microphone k, we observe a noisy speech signal

$$y_k(n) = (g_k * s)(n) + e_k(n) = x_k(n) + e_k(n) \tag{1}$$

where

- $g_k(n)$ is the impulse response from the source to microphone k
- $s(n)$ is the clean speech at microphone 1 (i.e., $g_1(n) = \delta(n)$),
- $e_k(n)$ is the noise (including interfering speech), and
- $x_k(n)$ is the clean speech signal recorded at microphone k.

In the frequency-domain, we have that

$$Y_k(\omega) = G_k(\omega)S(\omega) + E_k(\omega) = X_k(\omega) + E_k(\omega)\ . \tag{2}$$

Multi-channel Wiener Filtering
Mathematical model

We can collect all K microphone signals in a vector so that

$$\boldsymbol{Y}(\omega) = \begin{bmatrix} Y_1(\omega) \\ Y_2(\omega) \\ \vdots \\ Y_K(\omega) \end{bmatrix} = \begin{bmatrix} 1 \\ G_2(\omega) \\ \vdots \\ G_K(\omega) \end{bmatrix} S(\omega) + \begin{bmatrix} E_1(\omega) \\ E_2(\omega) \\ \vdots \\ E_K(\omega) \end{bmatrix} \tag{3}$$

$$= \boldsymbol{G}(\omega)S(\omega) + \boldsymbol{E}(\omega) \, . \tag{4}$$

Multi-channel Wiener Filtering

$$\boldsymbol{Y}(\omega) \longrightarrow \boxed{\boldsymbol{H}(\omega)} \longrightarrow \hat{S}(\omega)$$

We will extract $S(\omega)$ by designing a filter $\boldsymbol{H}(\omega)$ which

- filters out $\boldsymbol{E}(\omega)$ and
- does not change $S(\omega)$,

i.e.,

$$\hat{S}(\omega) = \boldsymbol{H}^H(\omega)\boldsymbol{Y}(\omega) = \underbrace{\boldsymbol{H}^H(\omega)\boldsymbol{G}(\omega)S(\omega)}_{\text{should be } S(\omega)} + \underbrace{\boldsymbol{H}^H(\omega)\boldsymbol{E}(\omega)}_{\text{should be } 0} . \qquad (5)$$

Multi-channel Wiener Filtering

If we define the objective

$$J(\boldsymbol{H}(\omega)) = \mathbb{E}\left[\left|S(\omega) - \boldsymbol{H}^H(\omega)\boldsymbol{G}(\omega)S(\omega)\right|^2\right] + \mathbb{E}\left[\left|0 - \boldsymbol{H}^H(\omega)\boldsymbol{E}(\omega)\right|^2\right], \tag{6}$$

its minimiser is the multi-channel Wiener filter given by

$$\hat{\boldsymbol{H}}_{\mathrm{MCWF}}(\omega) = \boldsymbol{\Phi}_{YY}^{-1}(\omega)\left([\boldsymbol{\Phi}_{YY}(\omega)]_{:,1} - [\boldsymbol{\Phi}_{EE}(\omega)]_{:,1}\right) \tag{7}$$

where

- $\boldsymbol{\Phi}_{YY}(\omega)$ is a matrix containing the cross PSDs of the microphone signals, and
- $\boldsymbol{\Phi}_{EE}(\omega)$ is a matrix containing the cross PSDs of the noise signals.

Multi-channel Wiener Filtering

Some comments:

- We do not need to know the array geometry (i.e., $\boldsymbol{G}(\omega)$) to implement the multi-channel Wiener filter.
- The multi-channel Wiener filter can cope with unsynchronized microphones to the extend that $\boldsymbol{G}(\omega)$ can absorb the synchronisation errors (i.e., clock offsets).
- The multi-channel Wiener filter can be implemented using distributed processing.
- While $\boldsymbol{\Phi}_{YY}(\omega)$ is easy to compute from the microphone signals,

$$\boldsymbol{\Phi}_{EE}(\omega) = \begin{bmatrix} \phi_{E_1 E_1}(\omega) & \cdots & \phi_{E_1 E_K}(\omega) \\ \vdots & \ddots & \vdots \\ \phi_{E_K E_1}(\omega) & \cdots & \phi_{E_K E_K}(\omega) \end{bmatrix} \tag{8}$$

is much harder.

Outline

Speech Presence Probability Estimation

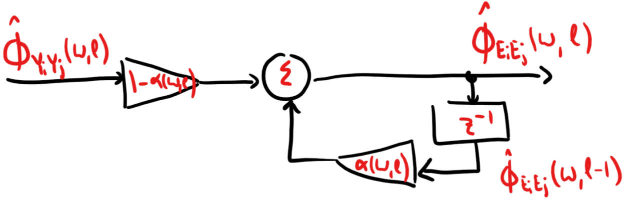
For microphone pair (i, j) in frame l, the noise cross PSD $\hat{\phi}_{E_iE_j}(\omega, l)$ can be estimated recursively using a first-order IIR filter as

$$\hat{\phi}_{E_iE_j}(\omega, l) = \alpha(\omega, l)\hat{\phi}_{E_iE_j}(\omega, l-1) + (1 - \alpha(\omega, l))\hat{\phi}_{Y_iY_j}(\omega, l) \tag{9}$$

where

$\hat{\phi}_{Y_iY_j}(\omega, l)$ is the cross PSD between microphone signals i and j, and

$\alpha(\omega, l)$ is the speech presence probability (SPP).



Speech Presence Probability Estimation

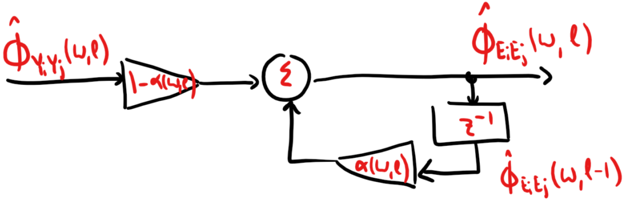
In a traditional voice activity detector (VAD), we make a hard decision.

H_0: There is no speech. Set $\alpha(\omega, l) = 0$ so that

$$\hat{\phi}_{E_i E_j}(\omega, l) = \hat{\phi}_{Y_i Y_j}(\omega, l) \tag{10}$$

H_1: There is speech. Set $\alpha(\omega, l) = 1$ so that

$$\hat{\phi}_{E_i E_j}(\omega, l) = \hat{\phi}_{E_i E_j}(\omega, l-1) \tag{11}$$



Speech Presence Probability Estimation

The SPP is a soft decision which we here define as

$$\alpha(\omega, l) = p(H_1(\omega, l)|\boldsymbol{Y}(\omega, l), \boldsymbol{\theta}(\omega, l)) \in [0, 1] \tag{12}$$

where

$p(\cdot)$ is a probability mass function

$\boldsymbol{\theta}(\omega, l)$ are model parameters (which are initially assumed known).

To compute the SPP, we have to be a little more explicit about our signal model.

Y. Zhao, J. K. Nielsen, J. Chen, and M. G. Christensen, *Model-based distributed node clustering and multi-speaker speech presence probability estimation in wireless acoustic sensor networks*, The Journal of the Acoustical Society of America 147, 4189-4201 (2020).

Speech Presence Probability Estimation

Signal model

Assume for microphone k that[1]

$$\begin{aligned} \text{No speech:} \quad & p(Y_k|\phi_{E_kE_k}, H_0) = \mathcal{CN}(0, \phi_{E_kE_k}) && (13) \\ \text{Speech:} \quad & p(Y_k|\phi_{E_kE_k}, \phi_{X_kX_k}, H_1) = \mathcal{CN}(0, \phi_{E_kE_k} + \phi_{X_kX_k}) \,. && (14) \end{aligned}$$

Note that

- this model is not completely consistent with the derivation of the multi-channel Wiener filter since it ignores correlation between the microphones, and
- the model parameters for microphone k are

$$\boldsymbol{\theta}_k(\omega, l) = \begin{bmatrix} \phi_{E_kE_k} & \phi_{X_kX_k} \end{bmatrix}^T \,. \quad (15)$$

[1] We sometimes omit the frequency and frame indices to simplify the notation.

Speech Presence Probability Estimation

Based on the model, we can calculate the likelihood ratio as

$$\mathcal{L}(Y_k|\boldsymbol{\theta}_k) = \frac{p(Y_k|\boldsymbol{\theta}_k(\omega, l), H_1)}{p(Y_k|\boldsymbol{\theta}_k(\omega, l), H_0)} \tag{16}$$

$$= (1+\xi_k)^{-1} \exp\left(\frac{|Y_k|^2}{\phi_{E_k E_k}} \frac{\xi_k}{\xi_k + 1}\right), \tag{17}$$

where ξ_k is the so-called a-priori SNR given by

$$\xi_k = \frac{\phi_{X_k X_k}}{\phi_{E_k E_k}} . \tag{18}$$

Speech Presence Probability Estimation

Since (also for H_0)

$$p(\boldsymbol{Y}|\boldsymbol{\theta}, H_1) = \prod_{k=1}^{K} p(Y_k|\boldsymbol{\theta}_k, H_1) \quad \text{and} \quad \mathcal{L}(Y|\boldsymbol{\theta}) = \prod_{k=1}^{K} \mathcal{L}(Y_k|\boldsymbol{\theta}_k) \quad (19)$$

it follows that so that the SPP can be calculated as

$$\alpha = p(H_1|\boldsymbol{Y}, \boldsymbol{\theta}) = \frac{p(H_1)\mathcal{L}(Y|\boldsymbol{\theta})}{p(H_0) + p(H_1)\mathcal{L}(Y|\boldsymbol{\theta})} \quad (20)$$

where $p(H_1)$ and $p(H_0)$ are priors. We can also easily incorporate multiple frames and multiple frequencies in the computation of the likelihood ratio!

Speech Presence Probability Estimation

So what did we observe?

- If we have a SPP, we can estimate the required noise cross PSDs to run the multi-channel Wiener filter.
- The SPP can be computed as
  1. compute the likelihood ratio for every channel
  2. combine the K likelihood ratios with prior probabilities.
- To compute the likelihood ratio in channel k, we have to know

$$\boldsymbol{\theta}_k(\omega, l) = \begin{bmatrix} \phi_{E_k E_k} & \phi_{X_k X_k} \end{bmatrix}^T . \tag{21}$$

  In practice, however, these PSDs are unknown and must be estimated from the data.

Outline

Single-channel Noise PSD estimation

Many methods exist for single-channel noise PSD estimation. The most well-known are

- minimum statistics (MS),
- improved minima controlled recursive averaging (IMCRA), and
- minimum mean squared error (MMSE).

These methods work well for stationary noise, but not for nonstationary noise.

- In our previous work, we developed a new model-based noise PSD estimator which works much better for nonstationary noise (e.g., bable noise).

Single-channel Noise PSD estimation
Model-based approach

Recall that we modelled the k'th microphone signal as

$$p(Y_k|\phi_{E_kE_k}, \phi_{X_kX_k}, H_1) = \mathcal{CN}(0, \phi_{E_kE_k} + \phi_{X_kX_k}) \tag{22}$$

where

$\phi_{E_kE_k}$ is the noise PSD, and

$\phi_{X_kX_k}$ is the speech PSD (as observed at microphone k) .

To estimate these, we do the following:

1. Assume that the clean speech and noise can be accurately modelled by autoregressive (AR) processes.
2. Pre-train codebooks of AR-vectors (i.e., parametrised speech and noise PSDs) that are typical for speech and for noise.
3. Form models of the microphone signal from all combinations of trained speech and noise PSDs.
4. Infer model parameters and probabilities and compute model averaged PSD estimates.

Single-channel Noise PSD estimation
Model-based approach

1. Autoregressive spectral modelling

- AR processes have been used extensively in speech coding.
- The PSD of an AR process is

$$\phi_{\mathrm{AR}}(\omega) = \frac{\sigma^2}{\left|1 - \sum_{p=1}^{P} a_p \exp(-j\omega p)\right|^2} = \sigma^2 \tilde{\phi}_{\mathrm{AR}}(\omega) \qquad (23)$$

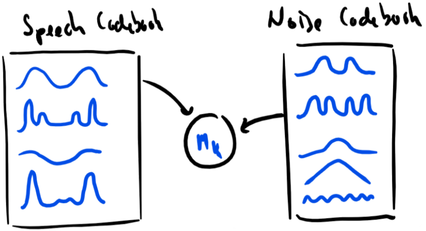
where

σ^2 is the excitation variance,
$\{a_p\}_{p=1}^{P}$ are the P AR parameters, and
$\tilde{\phi}_{\mathrm{AR}}(\omega)$ is the normalized AR spectrum (i.e., $\sigma^2 = 1$).

- For a low AR-order, the AR spectrum is smooth.

Single-channel Noise PSD estimation
Model-based approach

2. Training AR codebooks

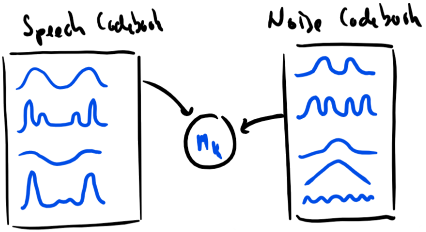


Typical AR-vectors of speech and noise can be obtained from training, and the results are stored in codebooks (Srinivasan et al., 2006, 2007). Both specific and general codebooks can be trained.

Single-channel Noise PSD estimation

Model-based approach

3. Modelling microphone signal using AR-codebooks



A model $\mathcal{M}_l$ is formed from a combination of AR-vectors in the speech and noise codebooks, i.e.,

$$p(Y_k|\sigma^2_{E_k}, \sigma^2_{X_k}, \mathcal{M}_l) = \mathcal{CN}(0, \sigma^2_{E_k}\tilde{\phi}^{(l)}_{E_k E_k} + \sigma^2_{X_k}\tilde{\phi}^{(l)}_{X_k X_k}) \,. \tag{24}$$

Note that $\sigma^2_{X_k}$ and $\sigma^2_{E_k}$ are the only unknowns in every model!

Single-channel Noise PSD estimation
Model-based approach

4. Infer model parameters and probabilities and compute PSDs

- For every model, the excitation variances $\sigma^2_{X_k}$ and $\sigma^2_{E_k}$ as well as the model probability $p(\mathcal{M}_l|Y_k)$ can be computed using a variational Bayesian algorithm.
- This algorithm also produces the PSD estimates $\hat{\phi}^{(l)}_{X_k X_k}$ and $\hat{\phi}^{(l)}_{E_k E_k}$ for every model.
- The final PSD estimates are given by

$$\hat{\phi}_{X_k X_k} = \sum_{l=1}^{L} p(\mathcal{M}_l|Y_k)\hat{\phi}^{(l)}_{X_k X_k} \tag{25}$$

$$\hat{\phi}_{E_k E_k} = \sum_{l=1}^{L} p(\mathcal{M}_l|Y_k)\hat{\phi}^{(l)}_{E_k E_k} \ . \tag{26}$$

Single-channel Noise PSD estimation

So what did we observe?

- To estimate the single-channel PSDs, we used autoregressive models trained on typical speech and noise data.
- This approach is advantageous since
  - it reduces the number of unknowns to just two model parameters in every frame (the excitation variances), and
  - it allows us to include prior information about typical speech and noise signals.
- The single-channel PSD estimates were computed by model averaging.
- We now finally have all the information we need to run our multi-channel Wiener filter!

Outline

Microphone Clustering

- Recall that we now for each microphone k have a speech and a noise model parametrized by AR paramaters.
- Idea: What if we use the single-channel models for clustering of the microphones?
- Then each cluster would contain mics observing the same speech and noise spectra!
- We can then use the clusters to perform enhancement of the closest speaker using a subnetwork.

Microphone Clustering

The Itakura-Saito divergence between spectra $\phi(\omega)$ and $\hat{\phi}(\omega)$ is defined as

$$D_{\text{IS}}(\phi(\omega), \hat{\phi}(\omega)) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{\phi(\omega)}{\hat{\phi}(\omega)} - \log \frac{\phi(\omega)}{\hat{\phi}(\omega)} - 1 d\omega. \tag{27}$$

Given the speech and noise models for each mic, k,

$$\hat{\phi}_{X_k X_k}(\omega) = \sum_{l=1}^{L} p(\mathcal{M}_l | Y_k) \hat{\phi}_{X_k X_k}^{(l)(\omega)} \quad \text{and} \quad \hat{\phi}_{E_k E_k}(\omega) = \sum_{l=1}^{L} p(\mathcal{M}_l | Y_k) \hat{\phi}_{E_k E_k}^{(l)(\omega)},$$

we propose to assign mics to clusters using the following divergence between the centroid of cluster c and microphone k:

$$D(c, k) = D_{\text{IS}}(\mu_{X_c X_c}(\omega), \hat{\phi}_{X_k X_k}(\omega)) + D_{\text{IS}}(\mu_{E_c E_c}(\omega), \hat{\phi}_{E_k E_k}(\omega)) \tag{28}$$

Microphone Clustering

The cluster c index set, $\mathcal{C}_c$, is defined as

$$\mathcal{C}_c = \{k | D(c,k) \leq D(b,k) \forall b\}, \quad (29)$$

and the centroid for the cluster is given by

$$\mu_{X_c X_c}(\omega) = \frac{1}{|\mathcal{C}_c|} \sum_{k \in \mathcal{C}_c} \hat{\phi}_{X_k X_k}(\omega), \quad (30)$$

and similarly for $\mu_{E_c E_c}$.
In words:

- the microphones are assigned to the cluster that has the closest centroid in the sense of the IS divergence.
- the centroid is simply the mean of the member of the cluster.

In practice, clustering does not need to be performed on a segment-by-segment basis.

Distributed Implementation

- The methods requires the following computations:
  - Finding the optimal speech and noise models for mic k (local).
  - Performing clustering based on these models (global).
  - Computing the SPP (global).
- The global computation problems can be solved using distributed consensus averaging methods.
- We here use the asynchronous PDMM (Zhang and Heusdens, 2017) to solve these problems efficiently!

Outline

Experimental Results

We will here present Experimental Results from the paper Y. Zhao, J. K. Nielsen, J. Chen, and M. G. Christensen *Model-Based Distributed Node Clustering and Multi-Speaker Speech Presence Probability Estimation in Wireless Acoustic Sensor Networks*, Journal of the Acoustical Society of America 147, 4189 (2020).

- Focuses on
  1. clustering nodes near speakers (distributed k-means with order estimation),
  2. computing an SPP for each cluster, and
  3. deriving algorithms for distributed processing (i.e. no fusion center)
- Comment: It uses different features and metrics for clustering than what I have described here and uses the Calinski-Harabasz criterion for finding the number of clusters.

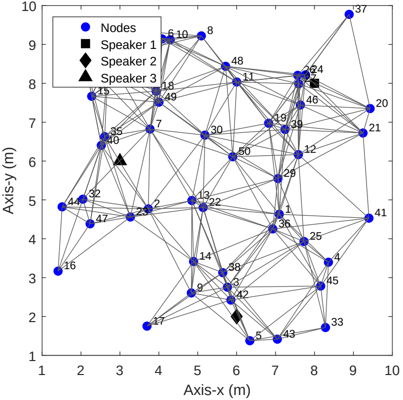
Experimental Results

Details:

- Simulated 10 m $\times$ 10 m $\times$ 3 m room with $T_{60} \approx 200ms$.
- We use 50 randomly placed mics with connections to other mics within 2.5 m.
- We use three speakers with the same power.
- Speech models trained on TIMIT and noise trained on AURORA.
- Testing is done on NOISEX-92 database.

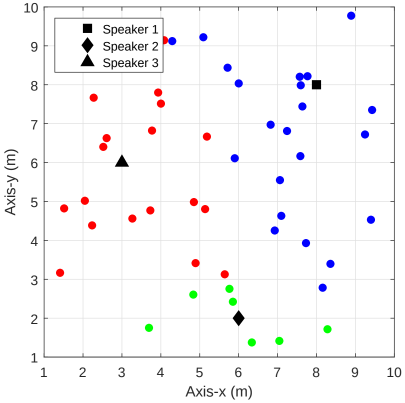
Experimental Results

Room setup



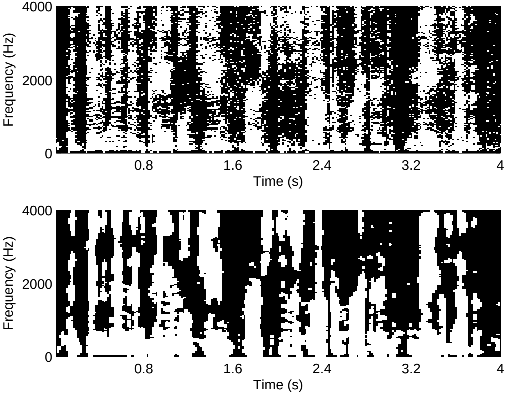
3 speakers and 50 microphones (nodes) in a dampened room ($T_{60} = 200$ ms) with background noise. Edges indicate transmission paths.

Experimental Results
Clustering example



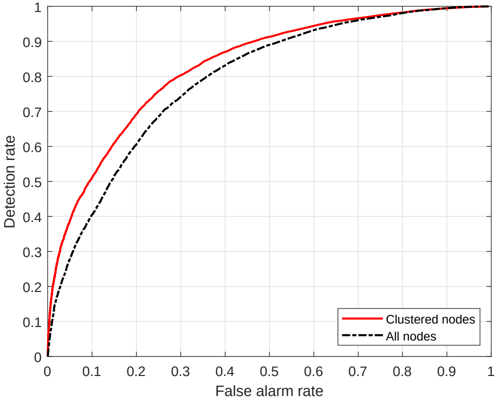
Background noise is babble noise at SNR of 10 dB.

Experimental Results
Detection example



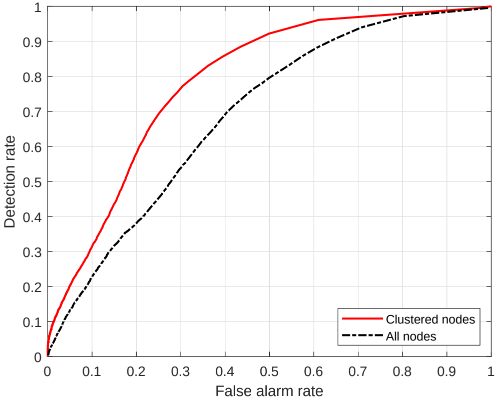
Ground truth (upper) and estimated (lower) voice activity detection (dark = absence and white = presence). False alarm rate is 0.2.

Experimental Results
Detection performance



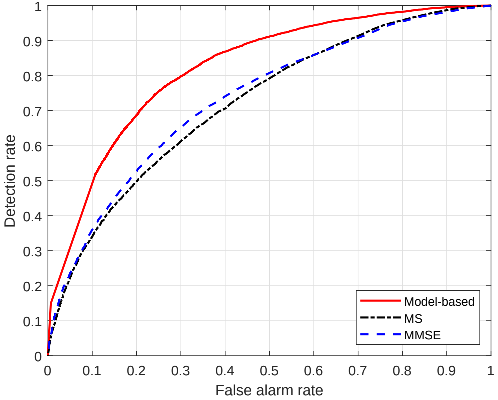
Speaker 1: It is better to use only the clustered microphones compared to all microphones.

Experimental Results
Detection performance



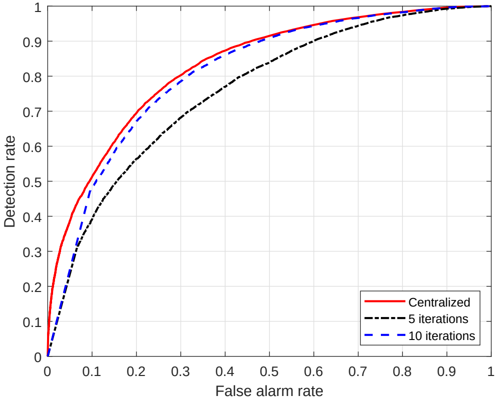
Speaker 2: It is better to use only the clustered microphones compared to all microphones.

Experimental Results
Noise PSD estimation methods



Results are for the microphones clustered around speaker 1 and babble noise at an SNR of 10 dB.

Experimental Results
Convergence of distributed algorithm



Results for one node as a function of the number of iterations of the PDMM algorithm.

Outline

Summary

- Speech enhancement algorithms such as the multichannel Wiener filter works for WASN.
- We need to know the noise cross PSDs to run, e.g., the multichannel Wiener filter.
- By using the concept of speech presence probabilities, we can estimate the noise cross PSDs from the single channel noise PSDs.
- We have shown how the single-channel noise PSD can be estimated using a model-based approach and that mics can be clustered based on the so-obtained models.
- Simulations results show that
  - clustering the microphones around the sources increases performance
  - the estimation of the SPP can be implemented using a distributed algorithm only requiring a few iterations

Questions?

AALBORG UNIVERSITY
DENMARK