Séparation de sources multicanale par machine learning

Stéphane KOMBO*,
Alexandre GUÉRIN†, Lauréline PEROTIN†, Srđan KITIĆ†

*TELECOM SudParis
†Orange Labs Cesson-Sévigné

Plan de l'exposé

I. Contexte
   A. Djingo, Smart Speaker d'Orange
   B. Traitement d'antenne
II. Localisation : Velocity Vector Module (VVM)
   A. Format ambisonique
   B. Algorithme, implémentation
III. Filtrage spatial : Séparation multicanale par machine learning
   A. Fonctionnement du système
   B. Optimisation du système

Bilan du stage

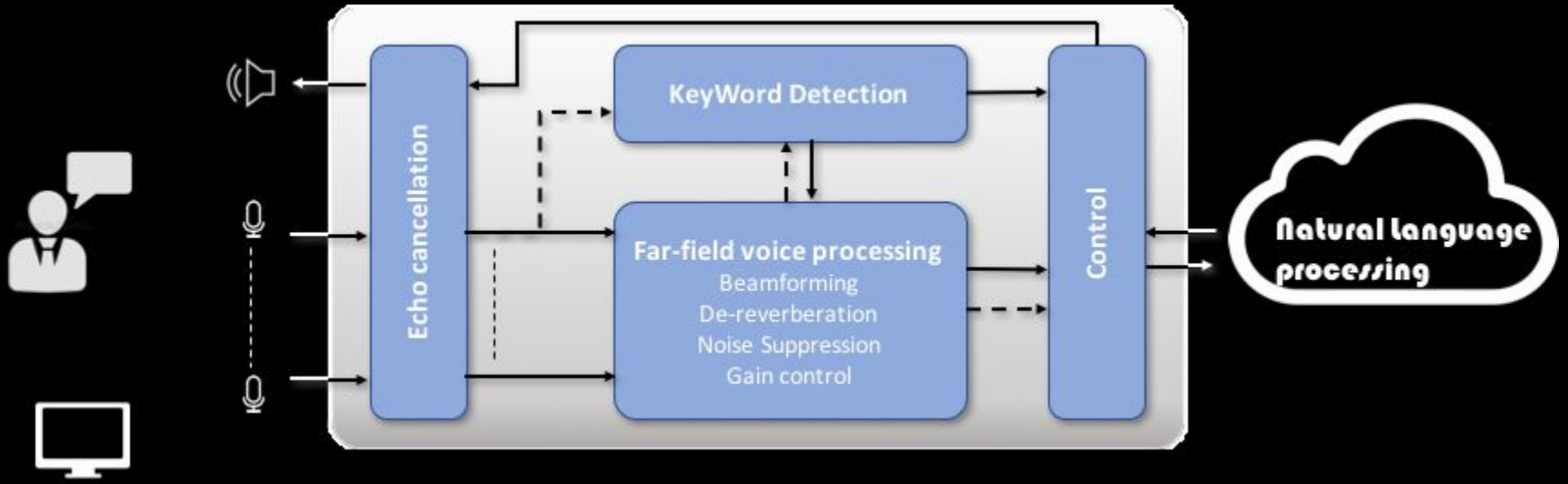
I. Contexte

Djingo

Le *Smart Speaker* d'Orange

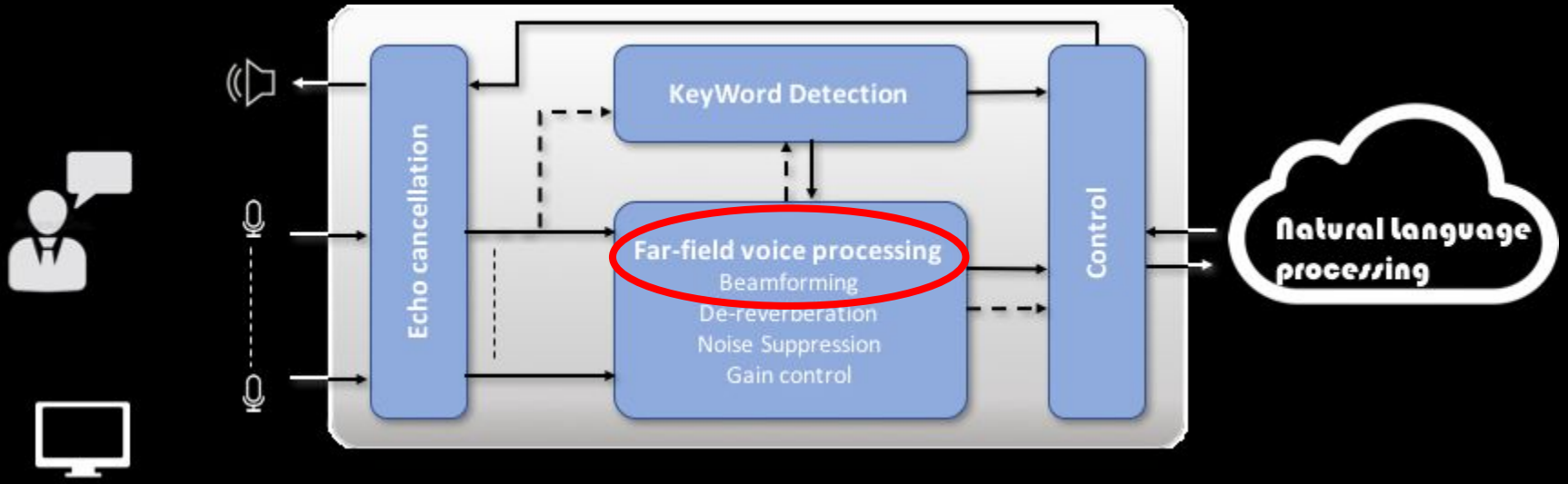
Smart Speaker

Enceinte connectée capable de dialoguer en “langage naturel”



Smart Speaker

Enceinte connectée capable de dialoguer en “langage naturel”



Prise de son “far-field”

Problématique : environnement bruyant et réverbérant

Nettoyage du signal
par traitement d’antenne :

- Localisation
- Filtrage spatial

II. Localisation : Velocity Vector Module

Format ambisonique

Une représentation multicanale

1. Intérêts de l’ambisonie
2. Format B

Format ambisonique

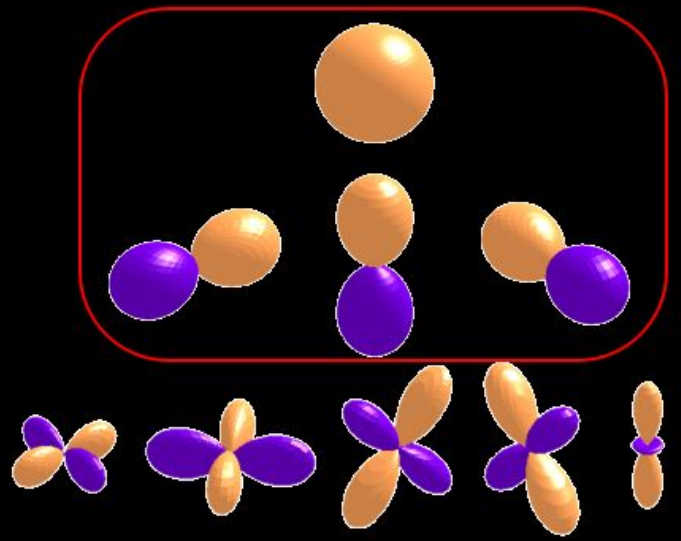
Une représentation spatiale
du champ acoustique

- Captation spatiale homogène
- Rendu indépendant
  du système de restitution
- Capacités à effectuer du
  traitement d'antenne :
  localisation, focalisation, ...

Format ambisonique

Format B : composantes d’ ordres 0 et 1

- Ordre 0 : composante omnidirectionnelle W
  → pression du champ sonore
- Ordre 1 : composantes directionnelles X, Y, Z
  → gradients du champ sonore ⇔ direction de propagation du champ sonore
- Ordres supérieurs : information directionnelle plus robuste



Algorithme, implémentation

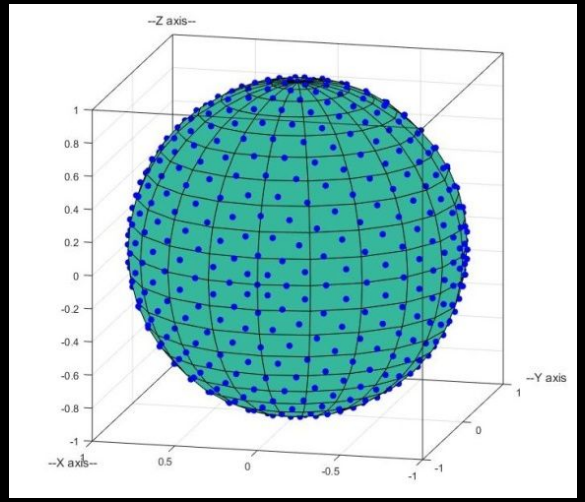
Localisation basée sur l'intensité acoustique

1. Intensité acoustique
2. Localisation et suivi des sources
3. Implémentation orientée objet

Intensité acoustique

Pour une source seule en milieu anéchoïque

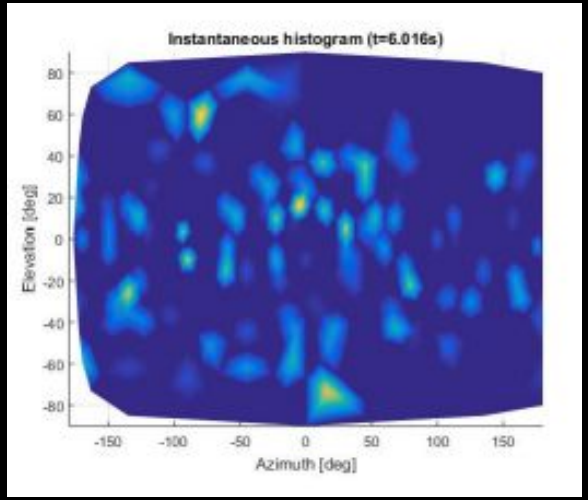
$$\begin{bmatrix} W \\ X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 1 \\ \cos\theta\cos\phi \\ \sin\theta\cos\phi \\ \sin\phi \end{bmatrix} p$$



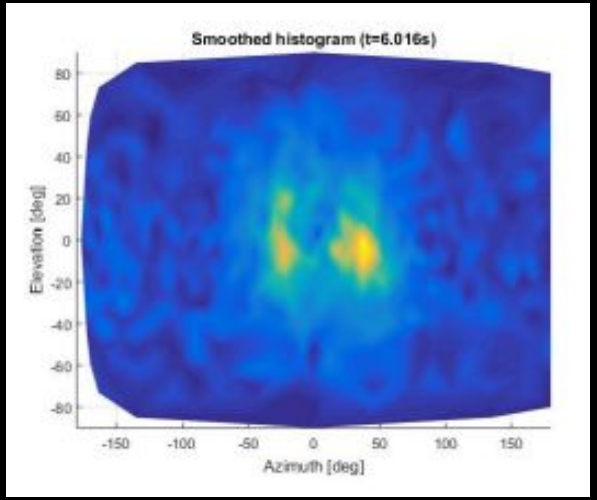
$$\vec{I}(t) = \begin{bmatrix} I_X(t) \\ I_Y(t) \\ I_Z(t) \end{bmatrix} = \begin{bmatrix} W(t)X(t) \\ W(t)Y(t) \\ W(t)Z(t) \end{bmatrix} = \begin{bmatrix} \cos\theta\cos\phi \\ \sin\theta\cos\phi \\ \sin\phi \end{bmatrix} p^2(t)$$

Localisation et suivi des sources

Analyse temps-fréquence dans le cas multisource



Histogramme pour une trame (40 ms)



Histogramme pour 50 trames (2 s)

Architecture du programme

Préparation des signaux
classe RawSignal()
Attributs : w, x, y, z, l
Méthode : __init__()

Localisation via les histogrammes
classe Quantized(RawSignal)
Attribut : index(θ, φ)
Méthodes : __init__(), histogram(), localisation()

Traitement complet	
fonction loc_multisource()	
Argument : signal ambisonique **s**	
Retour : localisation [θ(t), φ(t) for t in len(**s**)]	
Paramétrisation	Boucle principale
sound = RawSignal() sound.w(t), sound.x(t) sound.y(t), sound.z(t)	sound.l(t, f) sig = Quantized(sound) sig.localisation()

III. Filtrage spatial : Séparation multicanale par machine learning

Fonctionnement du système

Filtrage et *machine learning*

1. Filtre de Wiener multicanal
2. Estimation des matrices de covariance
3. La solution globale

Filtre de Wiener multicanal

Filtre en sous-bande optimal

- Mélange additif : $$\mathbf{x}(t, f) = \mathbf{s}(t, f) + \mathbf{n}(t, f)$$

Filtre de Wiener multicanal

Filtre en sous-bande optimal

- Mélange additif : $$\mathbf{x}(t,f) = \mathbf{s}(t,f) + \mathbf{n}(t,f)$$
- Filtrage linéaire : $$y(t,f) = \mathbf{w}(f)^H \mathbf{x}(t,f)$$

Filtre de Wiener multicanal

Filtre en sous-bande optimal

- Mélange additif : $\mathbf{x}(t,f) = \mathbf{s}(t,f) + \mathbf{n}(t,f)$
- Filtrage linéaire : $y(t,f) = \mathbf{w}(f)^H \mathbf{x}(t,f)$
- Critère : $\mathbf{w}(f) = \mathrm{argmin}_{\mathbf{h}}\, \mathbb{E}\{|s(t,f) - \mathbf{h}(f)^H \mathbf{x}(t,f)|^2\}$

Filtre de Wiener multicanal

Filtre en sous-bande optimal

- Mélange additif : $\mathbf{x}(t,f) = \mathbf{s}(t,f) + \mathbf{n}(t,f)$
- Filtrage linéaire : $y(t,f) = \mathbf{w}(f)^H \mathbf{x}(t,f)$
- Critère : $\mathbf{w}(f) = \mathrm{argmin}_{\mathbf{h}}\, \mathbb{E}\{|s(t,f) - \mathbf{h}(f)^H \mathbf{x}(t,f)|^2\}$
- Solution : $\mathbf{w}(f) = [\mathbf{\Phi}_{\mathbf{ss-r1}}(f) + \mathbf{\Phi}_{\mathbf{nn}}(f)]^{-1} \mathbf{\Phi}_{\mathbf{ss-r1}}(f)\mathbf{u}_0$

Filtre de Wiener multicanal

Filtre en sous-bande optimal

- Mélange additif : $\mathbf{x}(t,f) = \mathbf{s}(t,f) + \mathbf{n}(t,f)$
- Filtrage linéaire : $y(t,f) = \mathbf{w}(f)^H \mathbf{x}(t,f)$
- Critère : $\mathbf{w}(f) = \mathrm{argmin}_{\mathbf{h}}\, \mathbb{E}\{|s(t,f) - \mathbf{h}(f)^H \mathbf{x}(t,f)|^2\}$
- Solution : $\mathbf{w}(f) = [\mathbf{\Phi_{ss-r1}}(f) + \mathbf{\Phi_{nn}}(f)]^{-1} \mathbf{\Phi_{ss-r1}}(f)\mathbf{u}_0$

Estimation des matrices de covariances

Matrices de covariances de la forme :

$$\mathbf{\Phi_{ss}}(f) = \frac{1}{T}\sum_{t=0}^{T-1} \tilde{\mathbf{s}}(t,f)\tilde{\mathbf{s}}^{H}(t,f)$$

Estimation des matrices de covariances

Matrices de covariances de la forme :

$$\mathbf{\Phi_{ss}}(f) = \frac{1}{T}\sum_{t=0}^{T-1}\tilde{\mathbf{s}}(t,f)\tilde{\mathbf{s}}^{H}(t,f)$$

Estimation de **s** par masquage :

$$\tilde{\mathbf{s}}(t,f) = M_S(t,f)\mathbf{x}(t,f)$$

Estimation des matrices de covariances

Matrices de covariances de la forme :

$$\mathbf{\Phi_{ss}}(f) = \frac{1}{T}\sum_{t=0}^{T-1} \tilde{\mathbf{s}}(t,f)\tilde{\mathbf{s}}^{H}(t,f)$$

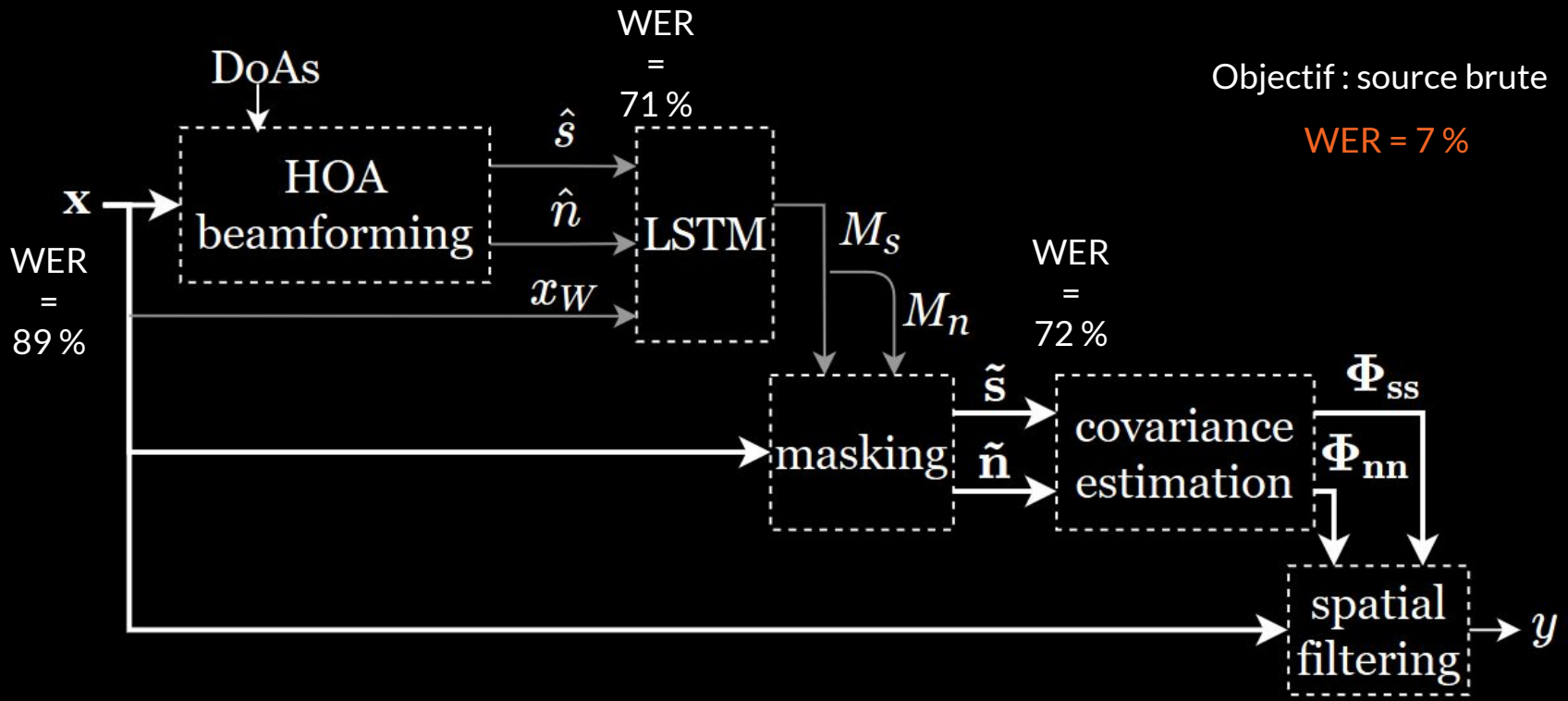
Estimation de **s** par masquage :

$$\tilde{\mathbf{s}}(t,f) = M_S(t,f)\mathbf{x}(t,f)$$

Masque estimé par *réseau de neurones* et *apprentissage supervisé :*

$$M_S(t,f) = \frac{|\mathbf{s}_W(t,f)|^2}{|\mathbf{s}_W(t,f)|^2 + |\mathbf{n}_W(t,f)|^2}$$

La solution globale



WER = 89 %

WER = 71 %

WER = 72 %

WER = 26 %

Objectif : source brute

WER = 7 %

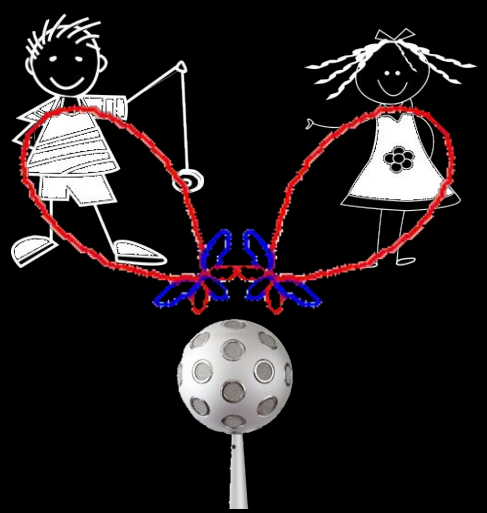
Optimisation du système

Synthèse des tests menés

1. Travaux sur les canaux d'entrée
2. Optimisation de l'apprentissage

Réseau initial

Entrées du réseau



- Beamformers

$$|\hat{s}(t,f)| = |\mathbf{b}_0^H \mathbf{x}(t,f)|$$

$$|\hat{n}(t,f)| = |\mathbf{b}_1^H \mathbf{x}(t,f)|$$

- Mélange omnidirectionnel

$$|\mathbf{x}_W(t,f)|$$

Signal	WER
Mélange omni	89 %
Beamformer	71 %
Masque prédit	72 %
Filtre spatial	**26 %**
Source cible	7 %

Entrées : rapports énergétiques

Idée : corréler entrée et sortie

- Masque à prédire :

$$M_S(t,f) = \frac{|\mathbf{s}_W(t,f)|^2}{|\mathbf{s}_W(t,f)|^2 + |\mathbf{n}_W(t,f)|^2}$$

- Rapports énergétiques :

$$\frac{|\hat{s}(t,f)|}{|\mathbf{x}_W(t,f)|}, \frac{|\hat{n}(t,f)|}{|\mathbf{x}_W(t,f)|}$$

- Beamformers : $|\hat{s}(t,f)|, |\hat{n}(t,f)|$

Signal \ WER	Rapports énergétiques	Réf.
Mélange omni	89 %	89 %
Beamformer	72 %	71 %
Masque prédit	78 %	72 %
Filtre spatial	**31 %**	**26 %**
Source cible	7 %	

Entrées : rapports énergétiques

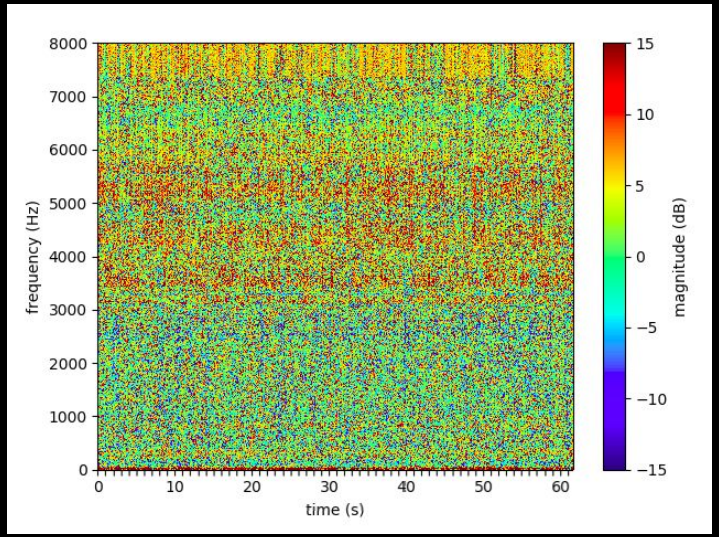
Idée : corréler entrée et sortie

- Masque à prédire :

$$M_S(t,f) = \frac{|\mathbf{s}_W(t,f)|^2}{|\mathbf{s}_W(t,f)|^2 + |\mathbf{n}_W(t,f)|^2}$$

- Rapports énergétiques :

$$\frac{|\hat{s}(t,f)|}{|\mathbf{x}_W(t,f)|}, \frac{|\hat{n}(t,f)|}{|\mathbf{x}_W(t,f)|}$$



- Beamformers : $|\hat{s}(t,f)|, |\hat{n}(t,f)|$

Réseau initial : normalisation canal par canal

$$\tilde{C}(seq, t, f) = \frac{C(seq, t, f) - \overline{m}_C(seq, f)}{\overline{\sigma}_C(seq, f)}$$

$$C \in \{|\tilde{\hat{s}}|, |\tilde{\hat{n}}|, |\tilde{\mathbf{x}}|\}$$

Inconvénient : casse la relation entre les canaux

Normalisation unique pour toutes les entrées

Idée : préserver les niveaux relatifs entre les entrées

$$\tilde{C}(seq, t, f) = \frac{C(seq, t, f) - \overline{m}(seq, f)}{\overline{\sigma}(seq, f)}$$

$$C \in \{|\tilde{\hat{s}}|, |\tilde{\hat{n}}|, |\tilde{\mathbf{x}}|\}$$

Statistiques calculées sur toutes les entrées

Signal \ WER	Normalisation unique	Réf.
Mélange omni	89 %	89 %
Beamformer	72 %	71 %
Masque prédit	73 %	72 %
Filtre spatial	**27 %**	**26 %**
Source cible	7 %	

Échelle : linéaire ou logarithmique

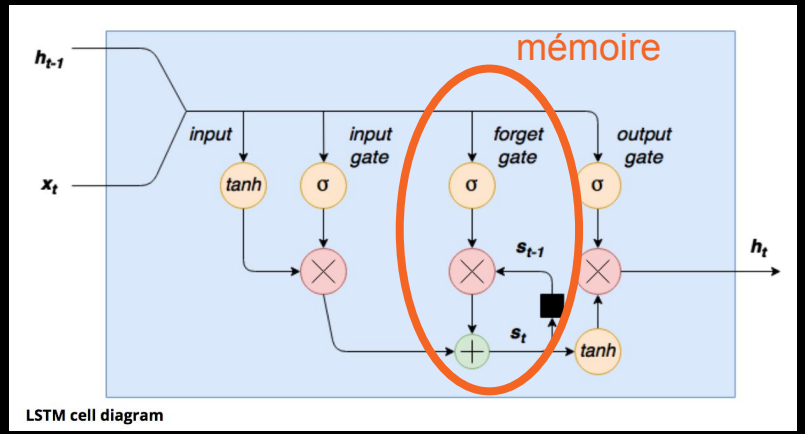
Idée : limiter les outlayers des entrées

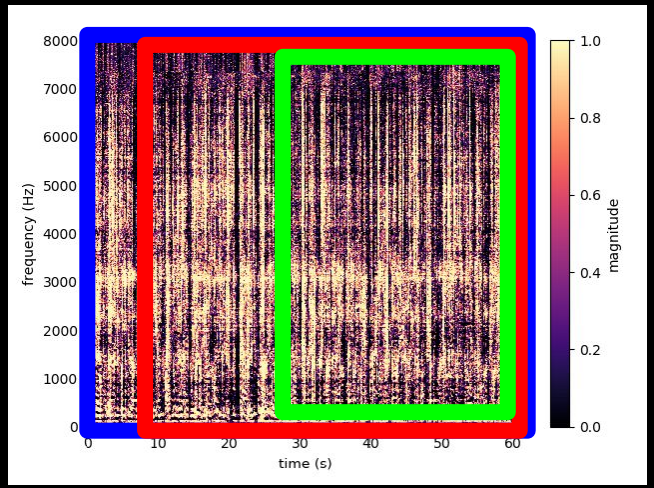
$$\log_{10} |\hat{s}(t,f)|$$
$$\log_{10} |\hat{n}(t,f)|$$
$$\log_{10} |\mathbf{x}_W(t,f)|$$

Signal \ WER	Logarithmique	Linéaire (Réf.)
Mélange omni	89 %	89 %
Masque prédit	77 %	72 %
Filtre spatial	**25 %**	**26 %**
Source cible	7 %	

Optimisation de l'apprentissage

$$\mathbb{E}_{seq,1\leq t\leq 25,f}\{|\widetilde{M}_S(seq,t,f) - M_S(seq,t,f)|^2\}$$





n = 0 n = 3 n = 12

$$\mathbb{E}_{n+1\leq t\leq 25}\{|\widetilde{M}_S(seq,t,f) - M_S(seq,t,f)|^2\}$$

Optimisation de l'apprentissage

$$\mathbb{E}_{n+1 \leq t \leq 25}\{|\widetilde{M}_S(seq, t, f) - M_S(seq, t, f)|^2\}$$

WER / Signal	n = 12	n = 3	n = 0 (Réf.)
Mélange omni	89 %	89 %	89 %
Beamformer	72 %	72 %	72 %
Masque prédit	74 %	70 %	71 %
Filtre spatial	**29.0 %**	**28.1 %**	**27.7 %**
Source cible	7 %		

Résultats :

- Pas d'amélioration
- Manque de données ?

Bilan du stage

Bilan du stage

Pour Orange

Une nouvelle version du VVM pour la localisation

- Une implémentation en Python
- Un code plus modulaire et évolutif

Travaux sur la séparation de sources complétés

- Une implémentation plus maniable et performante
- Une méthode d’apprentissage optimisée

Bilan du stage

Pour moi

Un expérience technique enrichissante

- Progrès en traitement du signal, machine learning
- Découverte de l'acoustique

Un début de carrière en Recherche et Développement

- Projet professionnel : ingénieur de recherche
- Thèse : traitement automatique de la parole en réunion

Merci

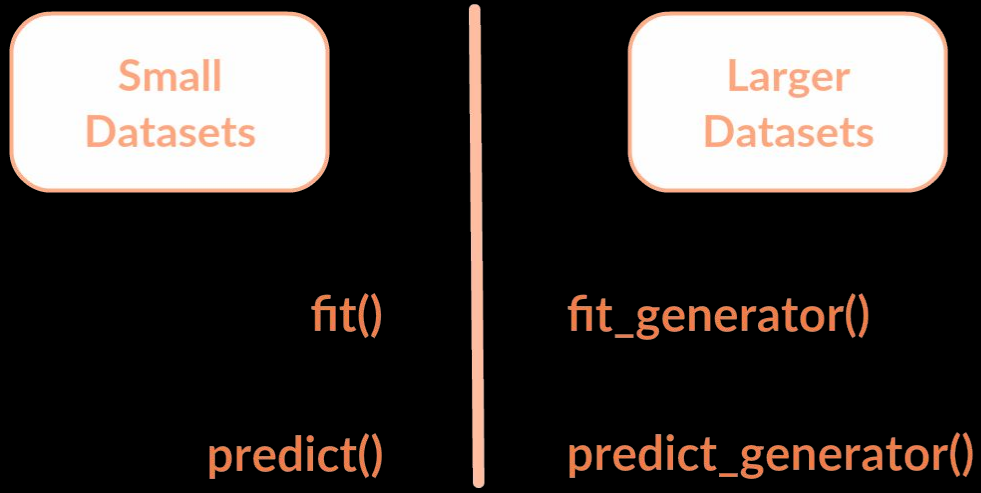
Des questions ?

Annexes

Optimisation de l'apprentissage

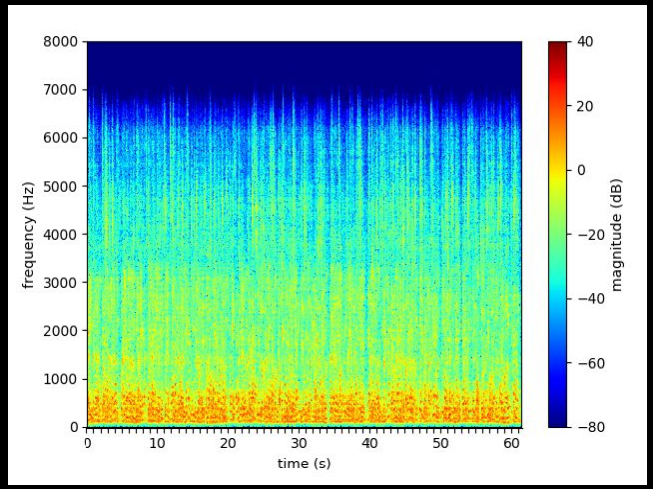
Classe DataGenerators de Keras

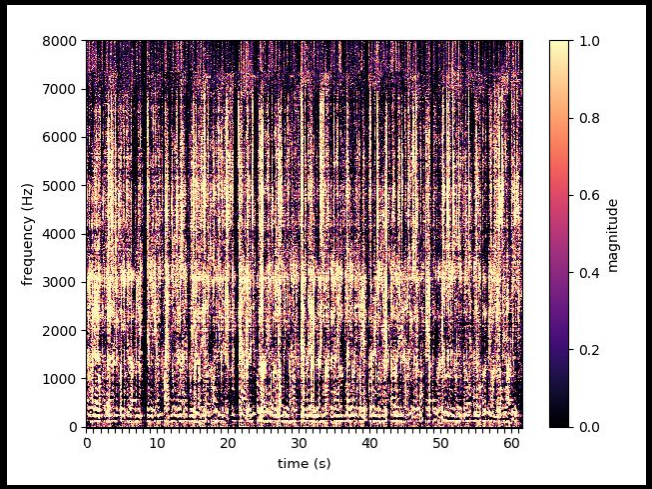
- Implémentation objet
- De meilleures méthodes pour l'apprentissage et la prédiction



Estimation des matrices de covariances

$\mathbf{x}(t, f)$ → $M_S(t, f)$

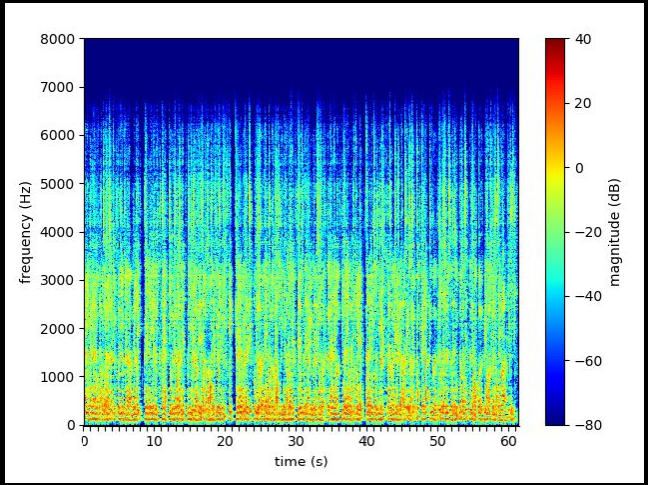


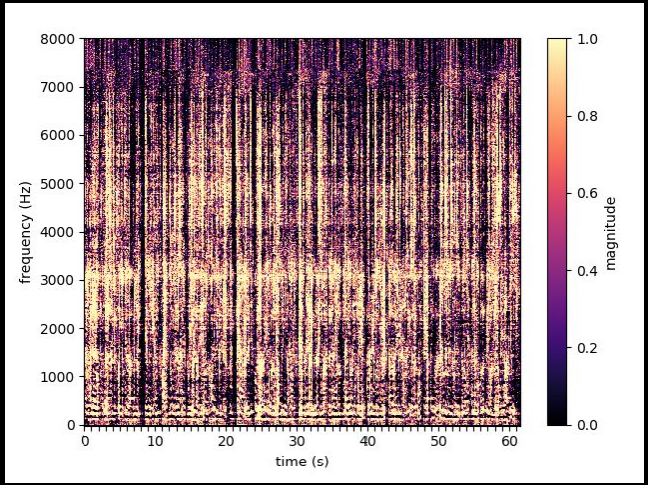


Masque estimé par *réseau de neurone*

Estimation des matrices de covariances

$$\tilde{\mathbf{s}}(t,f) = M_S(t,f)\mathbf{x}(t,f) \quad \longleftarrow \quad M_S(t,f)$$





Masque estimé par *réseau de neurone*