Modélisation de la parole par LPC

RAPPORT DE BE

Raphaël ADDI, Mehdi DHAHRI
INSA Toulouse – Département 3MIC
Encadrant : Pascal Noble

17 avril 2025

Table des matières

1 Introduction

Ce pré-rapport présente l'avancement du BE sur la modélisation de la parole à l'aide d'un modèle LPC (Linear Predictive Coding). L'objectif est d'analyser, synthétiser, et manipuler des signaux de parole à des fins de compression, reconnaissance et transformation vocale.

Tout d'abord il faut savoir ce qu'est le LPC nous allons le définir de facon simple et concise :

Le LPC, c'est une méthode pour analyser la voix en disant : "Chaque son qu'on prononce peut être deviné à partir des sons juste avant." Ça permet de reproduire ou compresser la parole sans stocker tous les sons, juste la façon dont ils changent.

2 Modèle mathématique de la production vocale

Un signal vocal $x[n]$ est modélisé comme la sortie d'un filtre AR (Auto-Régressif) d'ordre p :

$$x[n] = \sum_{k=1}^{p} a_k x[n-k] + e[n]$$

où $e[n]$ est le signal d'excitation :

— bruit blanc pour les sons non voisés,

— train d'impulsions (Dirac) pour les sons voisés.

3 Analyse LPC : du signal à l'encodage

3.1 Découpage en trames (fenêtrage)

Le signal est découpé en trames de $n_w = 240$ échantillons (soit 30 ms à 8kHz) avec un recouvrement $R = 0.5$.

— `CreateTrame` : génère les trames pondérées par une fenêtre de Hann. La fenêtre de Hann, c'est un outil qu'on utilise pour adoucir les bords d'un signal quand on le découpe en morceaux. Ça évite les sauts brutaux qui créent des erreurs dans les calculs de fréquences. Ca rend l'analyse du son plus propre et plus précise.

— `AddTrame` : Après avoir découpé le signal en morceaux on peut le reconstruire par Overlap-Add.

3.2 Estimation des coefficients LPC

Chaque trame x est modélisée comme une combinaison linéaire de ses valeurs passées. On cherche un vecteur $a = (a_1, \ldots, a_p)^T$ tel que :

$$x[t] \approx \sum_{k=1}^{p} a_k x[t-k] + e[t] \quad \text{pour } t = p, \ldots, n-1$$

On forme alors un système surdéterminé $Xa = b$, où :

$$X = \begin{bmatrix} x[p-1] & x[p-2] & \ldots & x[0] \\ x[p] & x[p-1] & \ldots & x[1] \\ \vdots & \vdots & \ddots & \vdots \\ x[n-2] & x[n-3] & \ldots & x[n-p-1] \end{bmatrix}, \quad b = \begin{bmatrix} x[p] \\ x[p+1] \\ \vdots \\ x[n-1] \end{bmatrix}$$

On note nb le nombre de trames. Si $p < nb$, alors le système est surdéterminé, ce qui signifie qu'il y a plus d'équations que d'inconnues. On résout ce système au sens des moindres carrés :

$$a = (X^T X)^{-1} X^T b$$

L'erreur $e = b - Xa$ est modélisée comme un bruit blanc, hypothèse raisonnable dans le cas voisé. Elle peut cependant être testée dans le cas non voisé.

La variance du bruit est estimée par :

$$\sigma^2 = \mathrm{Var}(b - Xa)$$

La fonction `EncodeLPC` applique ce traitement à chaque trame du signal et retourne :

— $A \in \mathbb{R}^{p \times nb}$, matrice contenant les coefficients de chaque trame ;

— $G \in \mathbb{R}^{nb}$, vecteur des variances de chaque trame.

3.3 Erreur de reconstruction selon l'ordre p

On évalue la fidélité de la reconstruction en calculant l'erreur entre le signal original et le signal reconstruit à l'aide du modèle LPC pour différents ordres p.

Les métriques utilisées sont :

— La norme L_1, moyenne des valeurs absolues des erreurs :

$$\|x - \hat{x}\|_{L_1} = \frac{1}{N} \sum_{n=0}^{N-1} |x[n] - \hat{x}[n]|$$

— La norme L_2, racine carrée de la moyenne des carrés des erreurs :

$$\|x - \hat{x}\|_{L_2} = \sqrt{\frac{1}{N} \sum_{n=0}^{N-1} (x[n] - \hat{x}[n])^2}$$

Ces erreurs sont calculées trame par trame, puis moyennées.

Ordre p	Erreur L_1	Erreur L_2
6	0.0807	0.1379
12	0.0797	0.1351
24	0.0788	0.1330

4 Synthèse LPC

La fonction `DecodeLPC` utilise les coefficients A et les variances G pour reconstruire un signal à partir d'un bruit blanc normalisé, en passant par la fonction `RunAR`.

Pour éviter que le signal reconstruit ne devienne trop fort ou déformé :

— un **clipping** est appliqué pour forcer les valeurs du signal à rester dans l'intervalle $[-1, 1]$;

— puis, une **normalisation à 90 %** de l'amplitude maximale est effectuée, ce qui signifie qu'on réduit un peu le volume du signal pour qu'il reste à 90 % de sa valeur la plus forte, afin d'éviter toute saturation lors de la lecture.

5 Synthèse croisée

La synthèse croisée consiste à appliquer les coefficients A d'un signal (modulation) sur les gains G d'un autre signal (porteur). Cela permet de transférer les caractéristiques spectrales d'une voix à une autre.

Fonctions principales :

— `EncodeLPC` pour calculer les coefficients et variances ;

— `RunAR` pour générer chaque trame ;

— `AddTrame` pour reconstruire le signal complet.

6 Vers la reconnaissance vocale

Après avoir modélisé et synthétisé le signal vocal à l'aide du modèle LPC, nous cherchons maintenant à exploiter ses paramètres pour identifier les sons prononcés. Cela nous conduit naturellement vers une première approche de reconnaissance vocale fondée sur l'analyse du spectre LPC et la comparaison avec un dictionnaire de sons connus. Nous détaillons ci-dessous les étapes de cette procédure.

6.1 Voisement et pitch

On commence par détecter les trames voisées (voisement) grâce à la fonction `Voisement()`, fondée sur l'autocorrélation.

La fonction `EstimationPitch()` estime ensuite la fréquence fondamentale (pitch) des trames voisées.

6.2 Excitation et synthèse améliorée

Selon le voisement, l'excitation est :

— un train d'impulsions pour les trames voisées ;

— du bruit blanc pour les trames non voisées.

La fonction `ConstructionExcitation()` crée ce signal d'excitation, qui est ensuite filtré par `SyntheseLPC()` pour obtenir un signal de parole synthétique.

6.3 Paramètres spectraux et dictionnaire

Pour chaque trame du signal, on extrait les `formants`, c'est-à-dire :

— les `positions des pics` du spectre LPC (fréquences caractéristiques),

— les `amplitudes` de ces pics (intensité spectrale).

Ces caractéristiques sont à la fois `stables` et `discriminantes` entre chaque sons (notamment pour les voyelles). Elles permettent donc de reconnaître ou différencier les sons parlés.

On les compare ensuite aux formants extraits d'un `dictionnaire de sons connus` (préconstruit à partir de fichiers `.wav` il y en a environ 70 afin d'avoir un maximum de comparaison possible si on voulait être plus précis ils nous en aurait fallut beaucoup plus) a l'aide d'une mesure de distance. Le son du dictionnaire ayant les formants les plus proches est considéré comme le plus ressemblant.

La fonction `EstimationSon(...)` permet d'automatiser ce processus :

— elle découpe le signal inconnu en trames,

— extrait les formants de chaque trame,

— les compare à ceux du dictionnaire,

— et retourne une séquence d'indices correspondant au son le plus proche pour chaque trame.

Cependant, des erreurs ponctuelles peuvent apparaître à cause du bruit ou de variations locales. Pour améliorer la cohérence temporelle, on utilise alors la fonction `CorrectionSon(...)`, qui :

— détecte les trames isolées incohérentes,

— les remplace par le son dominant dans les trames voisines

Cette post-correction permet de lisser la reconnaissance dans le temps et d'obtenir un résultat plus fiable.

7 Conclusion et perspectives

Le modèle LPC est efficace pour compresser, synthétiser et transformer des signaux vocaux. Il permet également d'extraire des caractéristiques utiles pour la reconnaissance vocale.

Perspectives :

— Améliorer la robustesse en environnement bruité ;

— Tester d'autres modèles d'excitation plus réalistes ;

— Développer une reconnaissance multi-locuteur avec apprentissage.

8 Annexe

Nous vous avons mis les fichiers `.wav` et les graphiques sur un GitHub afin que vous puissier tout retrouver.

Références

[1] Maitine Bergounioux, *Mathématiques pour le Traitement du Signal*, Éditions Ellipses, 2004.

[2] Hyung-Suk Kim, *Linear Predictive Coding is All-Pole Resonance Modeling*, arXiv preprint arXiv :1606.04035, 2016.