Universidade de Aveiro
2009

Departamento de Electrónica, Telecomunicações e Informática

Gonçalo Adelino Loureiro Miguel

Speech Enhancement para Reconhecimento de Fala


Universidade de Aveiro
2009
Departamento de Electrónica, Telecomunicações e Informática

Gonçalo Adelino Loureiro Miguel

Speech Enhancement para Reconhecimento de Fala

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Dr. António Joaquim da Silva Teixeira, Professor Auxiliar, e da Dra. Ana Maria Perfeito Tomé, Professora Auxiliar, ambos do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico este trabalho a todos os que sempre me apoiaram, família e amigos.

O júri

Presidente

Prof. Dr. Tomás António Oliveira e Silva
Professor associado da Universidade de Aveiro

Vogais

Prof. Dr. António Joaquim da Silva Teixeira
Professor auxiliar da Universidade de Aveiro (Orientador)

Prof. Dra. Ana Maria Perfeito Tomé
Professora associada da Universidade de Aveiro (Co-Orientadora)

Prof. Dr. Carlos Jorge da Conceição Teixeira
Professor Auxiliar da Faculdade de Ciências da Universidade de Lisboa

Agradecimentos

A todos os amigos que me acompanharam ao longo do meu percurso académico, estando sempre presentes em todas as ocasiões, contribuindo para a conclusão desta fase da minha formação académica.

À minha família, com especial relevo aos meus pais, que nunca me negaram apoio e sempre estiveram disponíveis.

Ao Prof. Dr. António Joaquim da Silva Teixeira e a Profa. Dra. Ana Maria Perfeito Tomé, pela disponibilidade demonstrada e prontidão na resolução dos problemas que foram surgindo ao longo desta dissertação.

Palavras-chave

Reconhecedores de fala, Speech Enhancement, Singular Value Decomposition (SVD), desempenho, ruído

Resumo

A utilização de reconhecedores de fala, em ambientes industriais e domésticos é, cada vez mais, uma constante. A presença de ruído é um dos factores com que nos debatemos, pois condiciona bastante o seu desempenho. Com a realização desta dissertação, pretende-se aplicar metodologias de Speech Enhancement baseadas em SVD, capazes de melhorar esta condicionante. Os sinais de teste são pré-processados com o bloco de Speech Enhancement, antes de serem processados pelos reconhecedores previamente treinados. Criaram-se reconhecedores de fala, dependentes do orador, para dois cenários de utilização distintos, controlo de cadeira de rodas e controlo de sala de cinema em casa. Nos resultados apresentados, o desempenho dos classificadores foi avaliado em condições diferentes, como adição de ruído e aplicação do bloco de Speech Enhancement, comparando-se percentagens de reconhecimento, que representam o número de palavras reconhecidas das tarefas a executar.

Keywords Speech recognizers, Speech Enhancement, Singular Value Decomposition (SVD), performance, noise

Abstract The use of speech recognizers in industrial and domestic environments has significantly grown in the last years. One of the issues that we face is the presence of noise, which severely degrades performance. The main goal of this work is to develop methodologies for Speech Enhancement based on SVD, capable of addressing this issue. Our test signals are pre-processed with a Speech Enhancement block, before being received by the previously trained recognizers. We have created two user-specific speech recognizers, for two distinct scenarios: control of a wheelchair and a cinema at home. In the results presented, we have evaluated the performance of the classifiers under different conditions, such as addition of noise and application of a Speech Enhancement block, by comparing the rates of recognition, which represent the number of recognized words for a specific task to be performed.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

EVD	*Eigen Value Decomposition*
HMM	*Hidden Markov Model*
HTK	*Hidden Markov Model Toolkit*
LS	*Least Squares*
MFCC	*Mel Frequency Cepstral Coefficients*
MLF	*Modified Least Squares*
MLS	*Master Label File*
MMF	*Master Macro File*
MV	*Minimum Variance*
SLF	*Standard Lattice Format*
SNR	*Signal Noise Ratio*
SSA	*Signal Subspace Approach*
TDC	*Time Domain Constraint*

Capítulo 1 **Introdução**

1.1. Motivação

Nas modernas aplicações de comunicação por voz, deixou de ser assumido que os sistemas desenvolvidos operam em ambientes favoráveis. No entanto, os sistemas de comunicação por voz estão constantemente a ser usados sob condições adversas, onde os seus utilizadores, mesmo assim, esperam receber serviços satisfatórios. Por esta razão, a investigação na implementação de métodos de Speech Enhancement tem sido intensificada, inovando nas estratégias a utilizar face aos desafios que vão surgindo.

Com a realização desta dissertação pretende-se a criação de métodos de Speech Enhancement capazes de melhorar o desempenho de reconhecedores de fala, em condições adversas, tal como elevado ruído.

Com este intuito, foram desenvolvidos e testados métodos SVD (métodos que permitem a separação do espaço do ruído do espaço do sinal utilizando técnicas baseadas no cálculo de componentes principais) que permitissem melhorar os reconhecedores de fala. Com a utilização destas técnicas seria possível extrair features do sinal em estudo, produzindo-se features robustas.

Uma vez que nos dias de hoje, cada vez mais as casas e outro tipo de espaços, se encontram equipados com sistemas automáticos que permitem a realização das mais variadas tarefas, a criação de métodos de Speech Enhancement permite uma acentuada evolução destes sistemas. Daí esta dissertação estimular a sua realização.

1.2. Objectivos

Na dissertação apresentada podemos destacar dois grandes objectivos. A implementação de um método de Speech Enhancement baseado em SVD e posterior aplicação do mesmo num sistema concreto, um reconhecedor de fala robusto e dependente do orador. Este trabalho não se remete à comparação entre vários métodos, mas sim a verificar o

desempenho de um dado método, quando aplicado a um cenário de utilização concreto. Por outro lado, a criação de reconhecedores não é o ponto fulcral do trabalho, a ideia é tentar obter melhor desempenho de um dado reconhecedor de fala, que possa já estar previamente treinado e, por aplicação de Speech Enhancement ao seu conjunto de teste, melhorar o seu desempenho.

1.3. Estrutura da Dissertação

A dissertação está dividida em 5 capítulos distintos, onde são relatadas algumas das técnicas de Speech Enhancement existentes, a forma como foram desenvolvidos os métodos em Matlab, a implementação do reconhecedor, os resultados obtidos com os testes realizados e, por fim, as conclusões retiradas deste projecto e o que poderá ser ainda acrescentado futuramente.

No primeiro capítulo, é referenciada a motivação existente para a realização desta dissertação, assim como os objectivos definidos para o mesmo.

Com o capítulo dois pretendeu-se apresentar de uma forma geral os métodos de Speech Enhancement mais relevantes, incluindo o utilizado nesta dissertação. Neste capítulo está descrito o conceito de Speech Enhancement, o modelo de sinal e de ruído, assim como algumas das técnicas subspace (separação do espaço do sinal do espaço do ruído) existentes.

O capítulo três contém a descrição das várias etapas realizadas. Destas podem enumerar-se a descrição do diagrama geral de todo o trabalho, os cenários definidos para o reconhecedor, o modo de implementação em Matlab das várias rotinas utilizadas (rotinas necessárias à extracção dos parâmetros MFCC e adição de ruído aos sinais áudio), os dois métodos de redução de ruído (Speech Enhancement e redução de ruído pelo processamento dos coeficientes mfcc e delta) e a sua implementação em Matlab.

O quarto capítulo aborda o conceito de Modelos de Markov (HMM) e descreve os passos necessários para a criação de um reconhecedor de fala, baseado em HTK e dependente do orador.

No quinto capítulo apresentam-se os resultados relativos à avaliação do reconhecedor criado. A sua avaliação será feita sob variadas condições, como a adição de ruído (branco e babble) e aplicação do método de Speech Enhancement implementado aos dois conjuntos de sinais áudio, treino e teste. As condições de teste do reconhecedor têm por objectivo verificar a necessidade de treinar o reconhecedor com adição de ruído e verificar a vantagem de utilização do método de Speech Enhancement implementado. São ainda apresentados resultados relativos a dois casos de estudo exploratórios, variação de limiar e redução de ruído pelo processamento dos coeficientes mfcc e delta.

Por fim, o sexto capítulo apresenta as conclusões retiradas consoante os resultados obtidos, assim como os possíveis melhoramentos/sugestões de continuação desta dissertação.

Capítulo 2 Speech Enhancement – Metodologias para melhoramento da qualidade de um sinal de voz degradado

2.1. Enquadramento geral

Como primeira abordagem ao tema, deverá ser definido o conceito de Speech Enhancement, o que ele representa e quais os objectivos da sua utilização.

Tal como o nome poderá indiciar, Speech Enhancement tem como definição o melhoramento da inteligibilidade e/ou qualidade de um sinal de voz degradado, recorrendo-se a ferramentas de processamento de sinal [11]. No entanto, existem duas razões mais significativas que tornam o Speech Enhancement um problema de difícil resolução.

Uma delas prende-se com a natureza e característica dos sinais de ruído, visto que estes podem mudar dramaticamente ao longo do tempo e de aplicação em aplicação. Por outro lado, torna-se complicado encontrar algoritmos versáteis e capazes de funcionar em diferentes contextos.

A segunda razão, também de importante relevo, relaciona-se com o desempenho medido, que pode ser definida de maneira diferente, consoante a aplicação em causa. Para determinar o desempenho recorre-se a dois critérios, qualidade e inteligibilidade, já referidos anteriormente.

A qualidade é um critério subjectivo, visto que fornece informação relativa à percentagem de palavras que podem ser correctamente identificadas por um grupo de ouvintes. Já na inteligibilidade é necessária atenção redobrada, pois quando o sinal em análise apresenta elevado ruído, o grupo de ouvintes poderá ter tendência a estar focado menos tempo na sua audição, provocando assim uma redução na inteligibilidade.

Como já foi referido no capítulo um, existe uma necessidade de constante inovação nas técnicas a implementar, de modo a tornar cada vez mais eficazes os sistemas de comunicação por voz em condições desfavoráveis.
Usualmente, as técnicas no domínio da frequência são as abordagens preferidas, devido a serem relativamente simples e de fácil implementação. Além disso, a voz é tradicionalmente analisada e compreendida no domínio da frequência, o que torna o processo neste domínio mais interessante e de mais fácil projecção.

Apesar de tudo, estes métodos, nomeadamente *spectral subtracion* e as suas variantes, não estão perto de oferecer total satisfação de soluções aos seus problemas inerentes, tais como: o ruído musical e o inevitável *trade-off* entre a distorção do sinal e o nível de ruído residual. No entanto, diferentes horizontes de pesquisa necessitam de ser investigados. Um dos potenciais domínios a ser utilizado nas operações de *subspace* é o *eigendomain*, como é o caso do método SVD utilizado na realização deste trabalho.

Em Speech Enhancement, as técnicas subspace (SSA) foram introduzidas por Dendrinos, que propôs a decomposição em valores singulares (SVD) de uma matriz de dados, de modo a remover o subespaço do ruído e posteriormente reconstruir o sinal pretendido a partir do subespaço do sinal [19]. Este tipo de técnica foi a base de desenvolvimento do método de Speech Enhancement implementado para a realização da dissertação. Toda a decomposição e processo associado á implementação da mesma, serão devidamente descritos no capítulo 3.

A utilização desta técnica ganhou mais ênfase a partir do momento que Ephraim e Van Trees propuseram uma nova variante, baseada na decomposição em valores próprios (EVD) da matriz de covariância do sinal de voz de entrada. De seguida, serão apresentados os modelos de sinal e de ruído, assim como decomposições diagonais e triangulares de matrizes, de modo a reduzir o subespaço deste último.

2.2. Modelo de Sinal e Ruído

Nesta secção, serão abordados os modelos de sinal e ruído e a sua representação por meio de decomposição de valores próprios (EVD) da matriz de covariância do sinal.

Primeiramente, deve ser definido o conceito de matriz de covariância. Esta matriz define-se como uma matriz simétrica que revela a covariância entre N variáveis. Por outro lado, uma matriz considera-se simétrica se a sua transposta for igual a ela própria, sendo este último conceito assumido quando existe troca de linhas por colunas de uma dada matriz.

De maneira a definir este conceito (covariância), é necessário também definir valor esperado (ou esperança) de uma dada variável aleatória discreta, como sendo a soma das probabilidades de cada possibilidade de saída da experiência, multiplicada pelo seu valor. Assim sendo, a esperança representa o valor médio esperado de uma experiência, se ela for repetida diversas vezes.

Por outro lado, a covariância entre duas variáveis aleatórias **X** e **Y**, com um dado valor esperado (ou esperança), $\boldsymbol{\varepsilon(X) = \mu_X}$ e $\boldsymbol{\varepsilon(Y) = \mu_Y}$, define-se por uma medida de como estas variam conjuntamente, sendo dada por $\boldsymbol{\mathrm{cov}(X,Y) = \varepsilon((X-\mu_X)(Y-\mu_Y))}$ **[18]**.

Se considerarmos um vector de elementos aleatórios $\boldsymbol{X = [X_1\ X_2\ \ldots X_N]}$, cada uma delas com uma variância finita, o **elemento genérico a_{ij}** da matriz de covariância será dado por:

$$\boldsymbol{a_{ij} = \mathrm{cov}(X_i, X_j) = \varepsilon\left[(X_i - \mu_i)(X_j - \mu_j)\right], \text{onde } \mu_i = \varepsilon(X_i) \text{ e } \mu_j = \varepsilon(X_j)} \quad \textbf{(2.1)}$$

Deverá ser tomado em conta também o conceito de variância de uma variável aleatória, que se define como uma medida da sua dispersão estatística, ou seja, indica de um modo geral a "distância" a que os seus valores se encontram do valor esperado.

Introduzidos os conceitos anteriores, passamos à descrição do modelo de sinal e de ruído já referido no inicio desta secção, necessário para a compreensão das decomposições matriciais posteriormente analisadas.

Para o modelo de sinal, serão considerados sinais estacionários de média zero e definidos como **vector s**. Este vector é apresentado sob a forma de coluna, cujos elementos pertencem a $\mathbb{R}^n$, com esperança nula, associando-se ainda a este um matriz de covariância **n x n**, dada por,

$$\mathbf{C_S} \backsimeq \boldsymbol{\varepsilon}\,(\mathbf{ss^T}) \tag{2.2}$$

No modelo de ruído, considera-se que a um sinal "limpo" $\in \mathbb{R}^n$, foi adicionado ruído (**vector e**) também $\in \mathbb{R}^n$, onde

$$\mathbf{s} = \bar{\mathbf{s}} + \mathbf{e} \tag{2.3}$$

O sinal "limpo", vector $\bar{\mathbf{s}}$, pertence a um subespaço finito definido por $\mathcal{s} \subset \mathbb{R}^n$, o qual é denominado por *signal subspace*.

A expressão 2.2 pode ser representada através das matrizes de covariância dos diferentes sinais enunciados, definindo-se por

$$\mathbf{C_S} = \mathbf{C_{\bar{s}}} + \mathbf{C_e} \tag{2.4}$$

A matriz de covariância do ruído, C_e, é *full rank*, ou seja, o número de linhas e colunas linearmente independentes é igual, isto é, não existe nenhum valor escalar que quando multiplicado por uma linha (ou coluna), seja possível obter outra linha (ou coluna) da mesma matriz. Sendo C_e *full rank*, então C_s também o será e o $rank(C_{\bar{s}}) = \dim(\bar{s}) = k < n$.

O principal objectivo dos métodos que serão apresentados na secção seguinte prende-se com a determinação da estimativa do sinal "limpo", denotando-se por $\hat{s}$. Para obter esta estimativa, o sinal de entrada deve ser tão longo quanto o pretendido, ou seja, deverá ser considerado um sinal $s' \in \mathbb{R}^N$, com N>n.

Definidos os dados do sinal s', estes são apresentados sob a forma de matriz de Hankel [9], de dimensão **m x n**, cujo formato é

$$\mathrm{H}(\mathrm{s}') = \begin{pmatrix} s'_1 & s'_2 & s'_3 & \cdots & s'_n \\ s'_2 & s'_3 & s'_4 & \cdots & s'_{n+1} \\ s'_3 & s'_4 & s'_5 & \cdots & s'_{n+2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ s'_m & s'_{m+1} & s'_{m+2} & \cdots & s'_N \end{pmatrix}$$

(2.5)

onde N = m + n – 1 e m ≥ n.

Dada a estrutura de dados definida, é possível expressar a matriz de covariância de **s** através de (2.5), obtendo-se

$$\mathbf{C_s} \approx \frac{\mathbf{1}}{\mathbf{m}} \mathbf{H^T H}$$

(2.6)

e, visto que $\mathbf{s}' = \bar{\mathbf{s}}' + \mathbf{e}'$, com $\bar{\mathbf{s}}'$ e $\mathbf{e}' \in \mathbb{R}^{\mathbf{N}}$, a expressão 2.3 pode ser reescrita, obtendo-se

$$\mathbf{H} = \bar{\mathbf{H}} + \mathbf{E}$$

(2.7)

com $\bar{\mathbf{H}} = \mathbf{H}(\mathbf{s}')$ e $\mathbf{E} = \mathbf{H}(\mathbf{e}')$ e, tal como já referido anteriormente, para $C_{\bar{s}}$, $\bar{H}$ tem *rank* igual a k.

Por fim, antes de passarmos à descrição dos métodos de determinação do sinal "limpo", é importante referir que após determinar a estimativa de $\overline{\mathrm{H}}$, ou seja, $\widehat{\overline{\mathrm{H}}}$, é necessário transformar esta matriz num vector de dados. Para realizar este processo, recorre-se ao cálculo da média das anti-diagonais de $\widehat{\overline{\mathrm{H}}}$, cuja descrição do mesmo está detalhada no capítulo 3, na secção onde se retrata a implementação do método de Speech Enhancement utilizado.

2.3. Decomposição matricial

Existem várias "categorias" no que respeita a metodologias de decomposição diagonal e triangular de matrizes, podendo-se destacar *SVD methods, rank-revealing triangular decompositions (UVT decompositions, Symmetric VSV decompositions), GSVD methods, Triangular decompositions*, entre outras [9]. Como apenas foi utilizada a metodologia SVD, será apresentado este método na secção seguinte, assim como os estimadores possíveis dentro deste método.

2.3.1. **Métodos SVD**

De maneira a introduzir o conceito de métodos SVD, será referido o caso ideal do ruído branco. A matriz de covariância é uma matriz identidade multiplicada por um escalar, definindo-se por

$$\mathbf{C_e} = \boldsymbol{\sigma}_{\mathbf{R}}{}^{\mathbf{2}}\mathbf{I_n} \tag{2.8}$$

onde $\boldsymbol{\sigma}_{\mathbf{R}}{}^{\mathbf{2}}$ representa a variância do ruído.

Por sua vez, a matriz de covariância do sinal "limpo" é representada pela decomposição em valores próprios dada por

$$\mathbf{C}_{\bar{\mathbf{s}}} = \overline{\mathbf{V}}\overline{\boldsymbol{\Lambda}}\overline{\mathbf{V}}^{\mathbf{T}} \tag{2.9}$$

onde os valores próprios são $\bar{\lambda}_{k+1} = \cdots = \bar{\lambda}_n = 0$, onde *k* representa o rank da matriz $C_{\bar{s}}$. A matriz de covariância do ruído do sinal ao qual foi adicionado ruído branco, define-se por

$$\mathbf{C_s} = \mathbf{C_{\bar{s}}} + \boldsymbol{\sigma_R}^2 \mathbf{I_n} \tag{2.10}$$

cujos vectores próprios são os mesmos, enquanto que os valores próprios se encontram "shiftados" por $\sigma_R{}^2$, ou seja, $\bar{\lambda}_i + \sigma_R{}^2$. Observando estas definições, conclui-se que, dada a variância do ruído $\sigma_R{}^2$ e a decomposição em valores próprios de C_s, a reconstrução de $C_{\bar{s}}$ poderá ser obtida subtraindo $\sigma_R{}^2$ aos *k* valores próprios de maior relevância da matriz C_s, inserindo o resultado na equação 2.9.

No entanto, quando o ruído **e** está próximo de ser ruído branco, é usado um estimador para determinar a variância do mesmo, correndo-se o risco de se obter valores próprios negativos aquando da subtracção da variância, tornando-se pouco eficiente a aplicação da reconstrução referida.

Um algoritmo funcional é obtido substituindo as matrizes de covariância por estimativas computacionais. Por razões de ordem computacional, é conveniente descrever este algoritmo sobre a forma de dois SVDs:

$$\overline{\mathbf{H}} = \overline{\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}} = (\overline{\mathbf{U_1}}\,\overline{\mathbf{U_2}}) \begin{pmatrix} \overline{\boldsymbol{\Sigma}_1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} (\overline{\mathbf{V_1}}\,, \overline{\mathbf{V_2}})^{\mathbf{T}} \tag{2.11}$$

$$\mathbf{H} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathbf{T}} = (\mathbf{U_1}\mathbf{U_2}) \begin{pmatrix} \boldsymbol{\Sigma}_1 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_2 \end{pmatrix} (\mathbf{V_1}\,, \mathbf{V_2})^{\mathbf{T}} \tag{2.12}$$

onde $\overline{U}, U \in \mathbb{R}^{mxm}$ e $\overline{V}, V \in \mathbb{R}^{nxn}$ têm colunas ortogonais e $\overline{\Sigma}, \Sigma \in \mathbb{R}^{nxn}$ são diagonais. Estas matrizes podem ser particionadas em $U_1 \in \mathbb{R}^{mxk}$, $\overline{V}_1, V_1 \in \mathbb{R}^{nxk}$ e $\overline{\Sigma}_1, \Sigma_1 \in \mathbb{R}^{kxk}$.

Note-se que com estes dois SVDs se pode imediatamente obter a decomposição em valores próprios através do produto entre as matrizes, uma vez que:

$$\overline{\mathbf{H}}^{\mathbf{T}}\overline{\mathbf{H}} = \overline{\mathbf{V}}\overline{\mathbf{\Sigma}}^{\mathbf{2}}\overline{\mathbf{V}}^{\mathbf{T}} \quad , \quad \mathbf{H}^{\mathbf{T}}\mathbf{H} = \mathbf{V}\mathbf{\Sigma}^{\mathbf{2}}\mathbf{V}^{\mathbf{T}}$$

(2.13)

O subespaço pretendido, ou seja, o subespaço do sinal "limpo" será dado por $\overline{\boldsymbol{s}} = \boldsymbol{\mathcal{R}}(\overline{\mathbf{V}}_{\mathbf{1}})$ e poderá ser obtido estimando o matriz do sinal "limpo" $\overline{\mathrm{H}}$, ou seja, $\widehat{\mathrm{H}}$.

De forma a ser possível o produto entre as matrizes já referido, define-se que as matrizes $\overline{\mathrm{H}}$ e E satisfaçam as seguintes condições

$$\mathbf{E}^{\mathbf{T}}\mathbf{E} = \boldsymbol{\eta}^{\mathbf{2}}\mathbf{I}_{\mathbf{n}} \quad \text{e} \quad \overline{\mathbf{H}}^{\mathbf{T}}\mathbf{E} = \mathbf{0}$$

(2.14)

sendo $\eta^2 = m{\sigma_R}^2$, que representa uma estimativa da variância do ruído em função do parâmetro *m*.

A primeira condição requer que as colunas de $(\sqrt{m}\,\eta)^{-1}E$ sejam ortogonais. A segunda implica que $m \geq n + k$. Assim obtem-se que

$$\mathbf{H}^{\mathbf{T}}\mathbf{H} = \overline{\mathbf{H}}^{\mathbf{T}}\overline{\mathbf{H}} + \boldsymbol{\eta}^{\mathbf{2}}\mathbf{I}_{\mathbf{n}}$$

(2.15)

Inserindo-se as equações SVDs 2.11 e 2.12 multiplicadas por *m*, temos a relação

$$(\mathbf{V_1}, \mathbf{V_2}) \begin{pmatrix} {\mathbf{\Sigma_1}}^{\mathbf{2}} & \mathbf{0} \\ \mathbf{0} & {\mathbf{\Sigma_2}}^{\mathbf{2}} \end{pmatrix} (\mathbf{V_1}, \mathbf{V_2})^{\mathbf{T}} = (\overline{\mathbf{V_1}}, \overline{\mathbf{V_2}}) \begin{pmatrix} \overline{{\mathbf{\Sigma_1}}^{\mathbf{2}}} + \boldsymbol{\eta}^{\mathbf{2}}\mathbf{I_k} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\eta}^{\mathbf{2}}\mathbf{I_{n-k}} \end{pmatrix} (\overline{\mathbf{V_1}}, \overline{\mathbf{V_2}})^{\mathbf{T}}$$

(2.16)

onde I_k *e* I_{n-k} representam matrizes identidade.

Existem vários algoritmos computacionais capazes de determinar $\widehat{H}$. Os algoritmos existentes são *least-squares estimate (LS), modified least-squares estimate (MLS), minimum variance estimate (MV) e time-domain constraint estimate (TDC)* [9].

A implementação de cada um destes algoritmos será descrita pormenorizadamente na secção seguinte. A formulação genérica que permitirá a utilização de todos eles é dada por:

$$\widehat{\mathbf{H}}_{\mathbf{svd}} = \mathbf{U_1 \Phi\, \Sigma_1 V_1}^{\mathbf{T}} \tag{2.17}$$

sendo Φ uma matriz diagonal denominada de matriz de ganho, que deverá ser determinada criteriosamente a partir da tabela apresentada de seguida [9].

Algoritmo	*Matriz de ganho Φ*
Least – Squares	I_k
Modified Least-Squares	$(I_k - \eta^2 {\Sigma_1}^{-2})^{1/2}$
Minimum Variance	$(I_k - \eta^2 {\Sigma_1}^{-2})$
Time-Domain Constraint	$\left(I_k - \eta^2 {\Sigma_1}^{-2}\right) * (I_k - \eta^2 (1-\lambda) {\Sigma_1}^{-2})^{-1}$
MLSA	$(I_k - (\eta^2)^{1/2} {\Sigma_1}^{-\frac{1}{2}})$

Tabela 1 – Matriz de ganho Φ

2.3.2. **Estimadores para a matriz de ganho Φ**

Na presente secção serão apresentados os diferentes estimadores existentes para a matriz de ganho Φ definida na secção anterior.

Como já referido, existem 4 tipos de estimadores principais e 1 alternativo: LS, MLS, MV, TDC e um outro método que se denominou de MLSA, visto ser uma alternância do método MLS.

As expressões que definem cada um destes métodos estão referenciadas na tabela 1, onde η^2 representa a estimativa da variância do ruído em função do parâmetro *m*, já referido anteriormente. Esta variável, em termos práticos, define-se como a média dos valores próprios não utilizados, uma vez que é definido um limiar que representa a percentagem de valores próprios a usar em relação à soma total destes.

De forma a tornar as expressões seguintes coerentes com a tabela 1, estas são apresentadas relacionando-se com os valores singulares. Estes definem-se como sendo a raiz quadrada dos valores próprios da matriz de Scatter do sinal que contém os dados de entrada.

Considere-se a matriz

$$\mathbf{H} = \mathbf{U_1 \Sigma V_1}^{\mathbf{T}} = \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{u_i \sigma_1 v_i}^{\mathbf{T}} \tag{2.18}$$

onde $\boldsymbol{u_i}$ e $\boldsymbol{v_i}$ representa cada uma das colunas *i* das matriz $\boldsymbol{U_1}$ e $\boldsymbol{V_1}$, respectivamente.

O *enhancement* do sinal de voz é efectuado modificando o perfil espectral do sinal na reconstrução, ou seja, calculando

$$\widehat{\mathbf{H}} = \mathbf{U_1 \Sigma \Phi V_1}^{\mathbf{T}} \tag{2.19}$$

sendo a matriz Φ uma matriz diagonal (secção anterior), que tem por objectivo modificar os valores próprios do sinal.

Na literatura [9], podem ser encontradas diferentes propostas para calcular os valores de Φ, que passaremos a enunciar de seguida.

1. Least-Squares

Nesta primeira proposta, a matriz de ganho Φ é dada pela matriz identidade I, ou seja, cada elemento desta matriz é igual a 1.

$$\mathbf{g_i = 1} \tag{2.20}$$

$$\mathbf{\Phi = \Phi_{LS} = I} \tag{2.21}$$

A expressão 2.18 pode ser redefinida por

$$\mathbf{\hat{H} = \sum_{i=1}^{K} u_i \sigma_i g_i v_i^t = \sum_{i=1}^{K} u_i \sigma_i (1) v_i^t = U_1 \Sigma I \, V_1^T = U_1 \Sigma V_1^T} \tag{2.22}$$

2. Minimum Variance

Neste caso, considera-se que cada elemento da matriz de ganho Φ se define por

$$\mathbf{g_i = 1 - \frac{\eta^2}{\sigma_i^2}} \tag{2.23}$$

onde η^2representa a variância do ruído e σ_i^2 os valores próprios.

Assim sendo, a matriz definida em 2.18 por ser reescrita, obtendo-se

$$\mathbf{\hat{H} = \sum_{i=1}^{K} u_i \sigma_i (1 - \frac{\eta^2}{\sigma_i^2}) v_i^T = U_1 \Sigma (I - \eta^2 \Sigma^{-2}) V_1^T} \tag{2.24}$$

A matriz de ganho Φ será dada por

$$\boldsymbol{\Phi = \Phi_{MV} = I - \eta^2 \Sigma^{-2}} \tag{2.25}$$

3. Modified Least Squares

Na presente proposta, cada elemento da matriz de ganho será dado por

$$\mathbf{g_i} = \frac{\sqrt{\boldsymbol{\sigma_i}^2 - \boldsymbol{\eta}^2}}{\boldsymbol{\sigma_i}} = \frac{\boldsymbol{\sigma_i}\sqrt{\mathbf{1} - \boldsymbol{\eta}^2 \boldsymbol{\sigma_i}^{-2}}}{\boldsymbol{\sigma_i}} = \sqrt{\mathbf{1} - \boldsymbol{\eta}^2 \boldsymbol{\sigma_i}^{-2}} \tag{2.26}$$

Assim

$$\widehat{\mathbf{H}} = \sum_{\mathbf{i=1}}^{\mathbf{K}} \mathbf{u_i} \boldsymbol{\sigma_i} \sqrt{\mathbf{1} - \boldsymbol{\eta}^2 \boldsymbol{\sigma_i}^{-2}}\, \mathbf{v_i}^{\mathbf{T}} = \mathbf{U_1 \Sigma (I - \eta^2 \Sigma^{-2})^{1/2} V_1}^{\mathbf{T}} \tag{2.27}$$

definindo-se a matriz de ganho por

$$\boldsymbol{\Phi = \Phi_{MLS}} = (\mathbf{I} - \boldsymbol{\eta}^2 \boldsymbol{\Sigma}^{-2})^{\mathbf{1/2}} \tag{2.28}$$

4. Time-Domain Constraint

No caso deste estimador, cada elemento da matriz de ganho Φ é definido por

$$\mathbf{g_i} = \frac{\mathbf{1} - \frac{\boldsymbol{\eta}^2}{\boldsymbol{\sigma_i}^2}}{\mathbf{1} - \frac{\boldsymbol{\eta}^2}{\boldsymbol{\sigma_i}^2}(\mathbf{1} - \boldsymbol{\lambda})} \qquad \mathbf{com\ 0 \leq \lambda \leq 1} \tag{2.29}$$

Obtém-se a matriz $\widehat{H}$

$$\widehat{\mathbf{H}} = \sum_{\mathbf{i=1}}^{\mathbf{K}} \mathbf{u_i \sigma_i}((\mathbf{1} - \frac{\boldsymbol{\eta}^{\mathbf{2}}}{\boldsymbol{\sigma}_{\mathbf{i}}{}^{\mathbf{2}}})(\mathbf{1} - \frac{\boldsymbol{\eta}^{\mathbf{2}}}{\boldsymbol{\sigma}_{\mathbf{i}}{}^{\mathbf{2}}}(\mathbf{1} - \boldsymbol{\lambda}))^{-\mathbf{1}})\mathbf{v_i}^{\mathbf{T}}$$
$$= \mathbf{U_1 \Sigma}(\mathbf{I} - \boldsymbol{\eta}^{\mathbf{2}}\boldsymbol{\Sigma}^{-\mathbf{2}})(\mathbf{I} - \boldsymbol{\eta}^{\mathbf{2}}(\mathbf{1} - \boldsymbol{\lambda})\boldsymbol{\Sigma}^{-\mathbf{2}})^{-\mathbf{1}}\mathbf{V_1}^{\mathbf{T}}$$

(2.30)

onde

$$\boldsymbol{\Phi} = \boldsymbol{\Phi}_{\mathbf{TDC}} = (\mathbf{I} - \boldsymbol{\eta}^{\mathbf{2}}\boldsymbol{\Sigma}^{-\mathbf{2}})(\mathbf{I} - \boldsymbol{\eta}^{\mathbf{2}}(\mathbf{1} - \boldsymbol{\lambda})\boldsymbol{\Sigma}^{-\mathbf{2}})^{-\mathbf{1}}$$

(2.31)

5. MLSA

Este último estimador foi criado com base no estimador *MLS* já referenciado, introduzindo-se uma pequena modificação.

Dado o sinal de entrada sob a forma de matriz de Hankel (H), para obter a projecção dos dados temos que

$$\mathbf{P} = \mathbf{HV}$$

(2.32)

Como a decomposição é SVD

$$\mathbf{P} = \mathbf{U\Sigma}$$

(2.33)

e o objectivo é retirar a cada valor da diagonal o valor estimado para o ruído

$$\mathbf{P_{modificado}} = \mathbf{U}(\boldsymbol{\Sigma} - \sqrt{\boldsymbol{\eta}^{\mathbf{2}}}\mathbf{I})$$

(2.34)

A estimativa de H pode ser obtida a partir da equação anterior por

$$\widehat{\mathbf{H}} = \mathbf{P_{modificado}}\mathbf{V^T} = \mathbf{P}(\mathbf{I} - (\boldsymbol{\eta}^{\mathbf{2}})^{\mathbf{1/2}}\boldsymbol{\Sigma}^{-\frac{\mathbf{1}}{\mathbf{2}}})\mathbf{V^T} \quad \textbf{(2.35)}$$

Capítulo 3 Speech Enhancement baseado em SVD para reconhecimento de fala robusto

3.1. Diagrama geral do sistema associado ao reconhecedor de fala robusto e dependente do orador

O esquema apresentado representa a proposta do sistema a implementar. Os blocos referidos na figura seguinte representam partes distintas do sistema, cuja função será retratada de seguida.

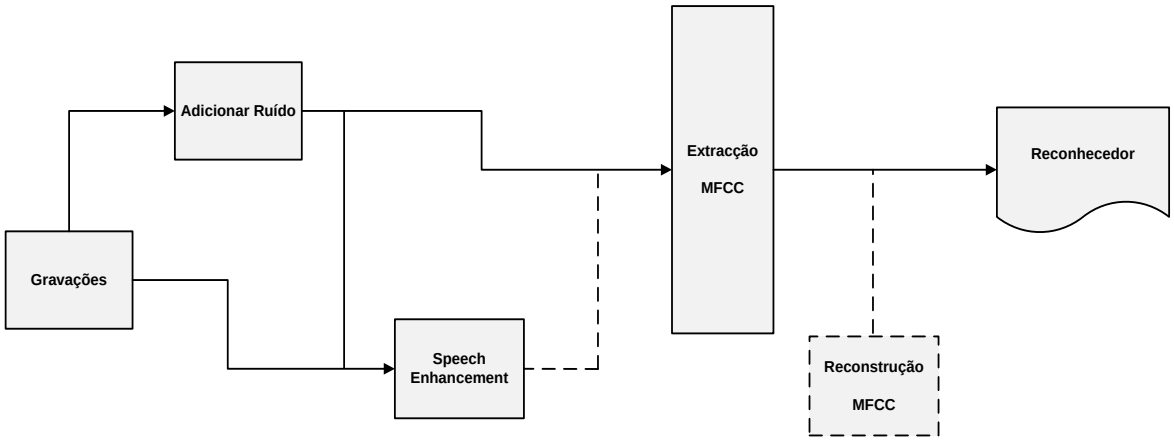

Figura 1 – Diagrama geral do sistema implementado (Speech Enhancement e Reconhecedor de fala)

No decorrer deste capítulo será apresentada a implementação do método de Speech Enhancement proposto, a extracção dos coeficientes MFCC das gravações criadas de acordo com o cenário pretendido para o reconhecedor e a adição de ruído a estas. A aplicação prática destes métodos prende-se com a criação de um *reconhecedor de fala dependente do orador* (capítulo 4), que permitirá avaliar o desempenho do mesmo nas mais variadas situações (com ou sem adição de ruído, com ou sem utilização do método de Speech Enhancement, etc).

Como será descrito de forma detalhada posteriormente, as gravações criadas têm por objectivo realizar tarefas relacionadas com um determinado ambiente/cenário (controlo de uma cadeira de rodas e controlo de uma sala de cinema em casa).

O sistema funcionará por blocos. As gravações são o primeiro bloco a referenciar, sendo o ponto de partida do sistema. Após a criação destas, é possível adicionar (ou não) ruído e aplicar (ou não) o método de Speech Enhancement implementado. O bloco seguinte remonta à extracção dos coeficientes MFCC, essenciais no processo, visto que estes revelam as características dos sinais de áudio criados (gravações). Extraídos estes parâmetros e criados os ficheiros que os contêm, procede-se à avaliação do reconhecedor previamente criado e treinado por um conjunto de gravações de treino.

Por outro lado, existe um bloco denominado de *reconstrução MFCC (caso exploratório)* em vez do método de Speech Enhancement, sendo possível fazer na mesma a avaliação do reconhecedor. Como já foi referenciado no capítulo anterior, após a extracção dos coeficientes MFCC é realizada a reconstrução baseada em SVD. De seguida, procede-se igualmente à avaliação do reconhecedor.

Para esta avaliação recorre-se a um conjunto de gravações de teste, as quais poderão sofrer o "efeito" provocado pelo bloco de ruído, de Speech Enhancement ou reconstrução MFCC, consoante o pretendido.

Com a criação deste último bloco pretende-se verificar a eficácia do método de Speech Enhancement (ou da reconstrução dos coeficientes MFCC) num dado cenário, alterando-se as condições das gravações, aproximando-as o mais possível do ambiente real.

3.2. Cenários exemplo para teste

Nesta secção estão referenciados os dois cenários de utilização do reconhecedor HTK criado, assim como as tarefas (comandos) que se pretendem realizar. Cada comando pode ser composto por uma ou mais palavras.

3.2.1. Controlo de cadeira de rodas

O primeiro cenário de teste do reconhecedor criado prende-se com o controlo de uma cadeira de rodas, sendo controlados os movimentos básicos para o seu funcionamento. Estes movimentos são baseados nas direcções possíveis de um joystick, tais como ***frente, direita, esquerda, parar, virar esquerda, virar direita, etc.***

3.2.2. Controlo de sala de cinema em casa

O segundo cenário tem por base o controlo de uma sala de cinema em casa. Segundo o definido, é possível realizar diversas tarefas de modo a tornar o ambiente semelhante ao de um cinema usual. Podemos destacar tarefas como ***ligar/desligar projector, subir/baixar estore, subir/baixar volume, subir/baixar tela, ligar/desligar colunas, ligar/desligar DVD, ligar/desligar luzes.***

3.3. Extracção dos parâmetros MFCC

A extracção dos parâmetros MFCC podia ser feita recorrendo ao HTK [1]. No entanto, um dos objectivos da dissertação era a criação de uma função Matlab que permite-se ter o mesmo desempenho do HTK. Para tal, procedeu-se à criação da mesma, tentando-se replicar o mesmo procedimento da ferramenta enunciada anteriormente. Nos passos seguintes serão enunciados todos os procedimentos desenvolvidos em Matlab, de forma a obter o sinal de entrada sob a forma de frame e posterior extracção dos referidos parâmetros.

As etapas determinantes [12] para a concretização deste objectivo definem-se por:

- Aplicação de pré-ênfase
- Aplicação da janela de análise
- Extracção dos parâmetros MFCC
- Energia e coeficientes delta

3.3.1. **Aplicação de pré-ênfase**

A aplicação deste processo ao sinal em estudo é de elevado relevo, visto que, quando falamos, o sinal de voz contém mais energia nas baixas do que nas altas frequências. Assim, será possível atenuar as componentes de baixa frequência do sinal, fenómeno denominado de pré-ênfase.

Para a implementação de pré-ênfase em processamento de sinal, procede-se à criação de um filtro digital de primeira ordem, cuja função de transferência é dada por

$$\mathbf{H(z) = 1 - az^{-1}}, \qquad \boldsymbol{com}\ \mathbf{0.9 \leq a \leq 1.0} \tag{3.1}$$

Aplicando a transformada inversa de Z, obtem-se a saída $y(n)$ em função da entrada $x(n)$, obtendo-se

$$\mathrm{y(n) = x(n) - a * x(n-1)} \tag{3.2}$$

onde $x(n)$ representa a amostra n do sinal de entrada e a assume os mesmos limites já definidos.

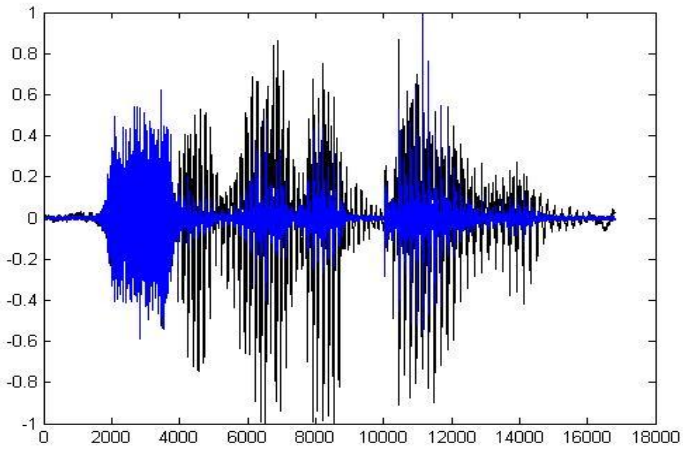

Figura 2 – Sinal Original (preto) e Sinal após pré-ênfase (azul)

3.3.2. **Aplicação da janela de análise**

A análise do sinal de entrada é feita em pequenas porções deste mesmo sinal, porções denominadas de *frame.* Visto que na aplicação prática estamos na presença de sinais não estacionários, ou seja, as suas características variam ao longo do tempo, consideram-se estas pequenas porções de 25ms, onde se assume que as características permanecem inalteradas.

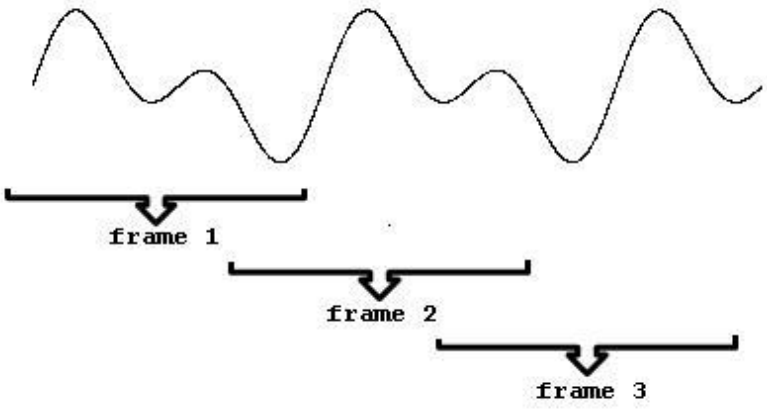

Figura 3 – Janela de análise (Hamming)

Para a obtenção de cada *frame*, é aplicada uma janela de Hamming de 25ms ao sinal de entrada, deslocando-se em incrementos de 10ms. Na expressão seguinte, podemos observar a definição da típica *janela de Hamming*

$$\mathbf{h(n)} = \begin{cases} \mathbf{0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right)} & \mathbf{para\ 0 \le n \le N-1} \\ \mathbf{0} & \mathbf{para\ outros} \end{cases} \qquad \mathbf{(3.3)}$$

onde N representa o número de amostras contidas dentro da janela, ou seja, o número de amostras existentes em 25ms do sinal de entrada.

3.3.3. **Extracção dos parâmetros MFCC**

De forma a extrair os coeficientes MFCC de um *frame* do sinal acústico de entrada, esta porção de sinal terá de "percorrer" várias etapas. Na figura seguinte podemos observar as diferentes etapas: FFT, banco de filtros triangular e aplicação de logaritmo [3].

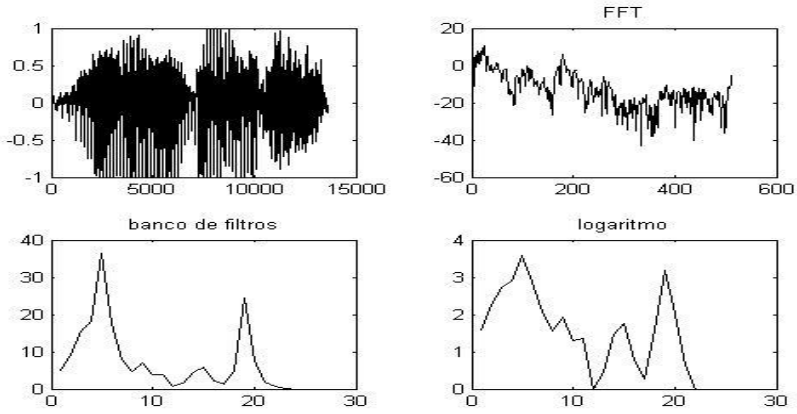

Figura 4 – Etapas para extracção dos coeficientes MFCC

A primeira etapa consiste na aplicação da *FFT* a um frame do sinal de entrada, de forma a obter a análise espectral do mesmo.

Seguidamente, o sinal passará por um banco de filtros triangular de maneira a ser modelado, uma vez que o ouvido humano não tem a mesma sensibilidade a todas as frequências. De maneira a conseguir este objectivo, cria-se um banco de filtros triangulares separados entre si segundo a escala de Mel. Esta escala foi proposta por Stevens, Volkman e Newman em 1937, e permite medir a sensação subjectiva do tom em função da frequência, definindo-se por

$$\mathbf{mel(f) = 1127 * ln\,(1 + \frac{f}{700})} \tag{3.4}$$

sendo f a frequência. Para $f < 500Hz$ a escala de mel é linear, enquanto para valores superiores é logarítmica.

Finalmente, na terceira etapa, como a resposta do ouvido humano ao sinal de voz é logarítmica, a melhor forma de modelar esta característica será aplicando a função logaritmo a cada um dos coeficientes MFCC determinados, sendo os mais relevantes os 13 primeiros.

3.3.4. **Energia e coeficientes delta**

O cálculo da energia do sinal é um dado muito importante, visto que as vogais e as sílabas apresentam diferentes valores energéticos quando comparados com as pausas. A energia, dado o sinal de voz x, pode ser obtida por

$$\mathrm{E} = \sum_{\mathrm{n}=0}^{\mathrm{N}-1} \mathrm{x}^2(\mathrm{n}) \tag{3.5}$$

Representando **N** o número de amostras total e **n** cada amostra **(1,2,3…N)** do sinal x.

Os coeficientes delta [12] permitem-nos obter informação sobre os coeficientes MFCC. Existem coeficentes delta de primeira e segunda ordem. Os primeiros retratam os coeficientes MFCC no que respeita á velocidade, enquanto os segundos referem a aceleração destes.

Para determinar os coeficientes delta recorre-se às expressões seguintes:

$$\mathbf{d_t} = \frac{\sum_{\theta=1}^{\Theta} \boldsymbol{\theta}(\mathbf{c_{t+\theta}} - \mathbf{c_{t-\theta}})}{\mathbf{2}\sum_{\theta=1}^{\Theta} \boldsymbol{\theta}^{\mathbf{2}}} \tag{3.6}$$

onde Θ representa o número total de coeficientes MFCC a processar e t o índice do coeficiente delta de primeira ordem calculado.

$$\mathbf{a_t} = \frac{\sum_{\theta=1}^{\Theta} \theta(\mathbf{d_{t+\theta}} - \mathbf{d_{t-\theta}})}{2\sum_{\theta=1}^{\Theta} \theta^2}$$

(3.7)

onde Θ representa o número total de coeficientes delta de primeira ordem a processar e t o índice do coeficiente delta de segunda ordem calculado.

A figura seguinte ilustra os coeficientes delta de primeira e segunda ordem (velocidade e aceleração) obtidos no processamento de um dado frame.

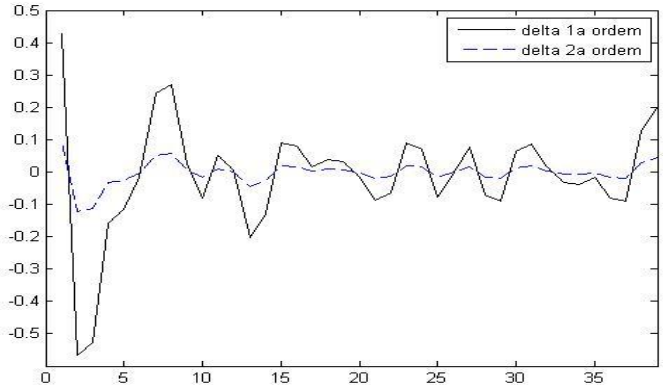

Figura 5 – Coeficientes delta de primeira e segunda ordem

3.4. Adição de ruído

Para a realização dos testes necessários à avaliação do reconhecedor, ocorreu a necessidade de adicionar ruído aos sinais áudio, treino e teste, recriando o mais fielmente possível o ambiente real de utilização de todo o sistema.

Para criar este efeito foram usadas duas formas distintas, para o ruído branco e ruído babble.

A adição de ruído branco foi feita recorrendo ao comando *awgn* do Matlab [14], cujos parâmetros de entrada são:

- Sinal ao qual se pretende adicionar ruído
- Nível de SNR
- Energia do sinal de entrada

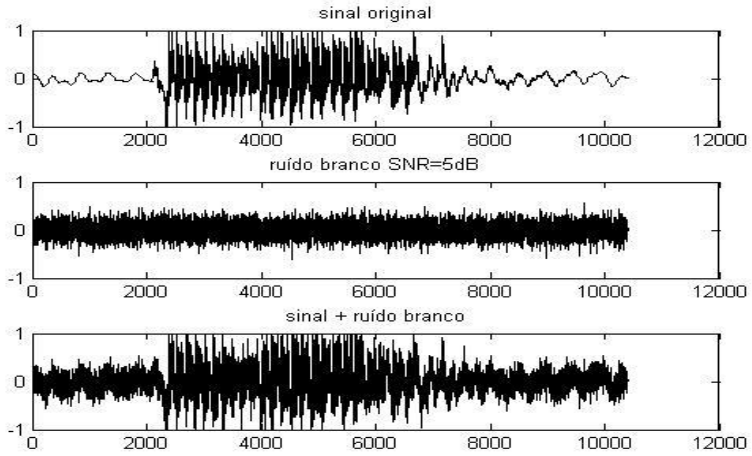

Figura 6 – Adição de ruído branco

Para adição de ruído babble, como o Matlab não possuí nenhuma ferramenta que o permitisse de forma automatizada, recorreu-se à base de dados AURORA [10] para obtenção de ruído babble com diferentes níveis de SNR (0,5,10 e 15). De forma a adicionar estes mesmos sinais de ruído, utilizou-se uma função Matlab já implementada por esta base, denominada de *addNoise* [10].

Como parâmetros de entrada desta função temos:

- Sinal ao qual se pretende adicionar ruído
- Sinal de ruído (neste caso babble noise)
- Nível de SNR
- Tamanho do sinal resultante (entrada + babble noise)

Na figura seguinte é possível observar 3 tipos de sinais definidos: o sinal original (entrada), o ruído babble e a resultante da adição dos dois anteriores, recorrendo-se á função Matlab referida.

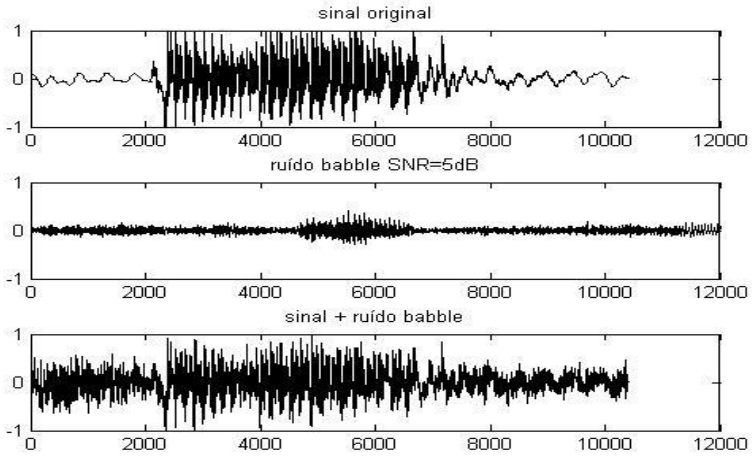

Figura 7 – Adição de ruído babble

3.5. Redução do ruído

Com o objectivo de reduzir o ruído existente no sinal de entrada, definiram-se duas metodologias diferentes. A primeira, o método de Speech Enhancement já enunciado no capítulo 2, consiste na eliminação do ruído no sinal de entrada, antes do cálculo dos coeficientes MFCC e Delta. A segunda, a qual se denominou de redução pelo processamento dos coeficientes MFCC e Delta, caracteriza-se pela eliminação do ruído nos parâmetros característicos extraídos.

3.5.1. **Redução por Speech Enhancement**

Na presente secção apresenta-se o método de Speech Enhancement implementado, baseado em SVD.

Na implementação deste método foram definidas condições iniciais [9], onde o sinal de entrada é apresentado segundo a equação 2.5, com **n = 30.** Sendo o sinal analisado em frames com duração de 25ms, e dada a frequência de amostragem deste, o número de amostras de cada frame é 400, ou seja, **N = 400.** Como já referido em 2.5, **N = m + n – 1**, obtendo-se **m = 371.**

Na implementação realizada, houve a necessidade de criação de várias rotinas de apoio ao método de Speech Enhancement pretendido, como é o caso da matriz de Hankel, Scatter e Kernel. Estas duas últimas obtêm-se através da multiplicação da matriz de Hankel pela sua matriz transposta e vice-versa.

Sendo o método utilizado SVD, as matrizes de Scatter e Kernel são úteis por permitirem obter as matrizes U, V e Σ, podendo os dados iniciais ser decompostos sob a forma de

$$\mathbf{H = U\Sigma V^T} \tag{3.8}$$

Recorrendo-se à matriz de Scatter dada por $S = H^T H$, obtem-se a matriz V contendo os vectores próprios de S e a matriz Σ, cuja diagonal contem os valores próprios de S.

O trecho de código seguinte demonstra a criação da matriz de Scatter e as respectivas matrizes V e Σ, em Matlab.

```
%Matriz de Scatter
inSCT = scatter_matrix(inHNK);

%Cálculo de V e Σ
[V,D]=eig(inSCT);
```

Após determinadas as matrizes Σ e V, procedemos à determinação do vector de projecção dos dados.

Na matriz Σ existem valores próprios que por terem um valor em módulo pequeno, aproximadamente zero, são pouco significativos. Como tal, na implementação do método

foi definido um valor limiar, que corresponde à percentagem em relação à soma total dos valores próprios, do número destes a utilizar na projecção dos dados e, consequentemente, na reconstrução. Considerando que os valores próprios são $\mathbf{d_1} > \mathbf{d_2} > \mathbf{d_3}$ ... $> \mathbf{d_n}$, obtem-se a expressão:

$$\frac{\mathbf{d_1} + \mathbf{d_2} + \mathbf{d_3} + \cdots + \mathbf{d_l}}{\mathbf{d_1} + \mathbf{d_2} + \mathbf{d_3} + \cdots + \mathbf{d_n}} * \mathbf{100} > \mathbf{LIMIAR} \tag{3.9}$$

onde $\boldsymbol{l} < \boldsymbol{n}$ e $\boldsymbol{d}$ representam os valores próprios da matriz Σ.

Definido o limiar, é calculado o número de valores próprios necessários para perfazer a percentagem pretendida pelo utilizador. Surge então uma nova matriz denominada de V_1, cujos valores próprios correspondem aos $\boldsymbol{l}$ escolhidos anteriormente.

A projecção dos dados pode ser descrita pela expressão:

$$\mathbf{P} = \mathbf{H}\mathbf{V_1} \tag{3.10}$$

onde H representa a matriz sob a forma de 2.5 e V_1 a matriz V reestimada, com $\boldsymbol{l} < \boldsymbol{n}$ colunas.

Para reconstruir a matriz $\widehat{H}$, para além das matrizes V_1 e P, é necessário introduzir uma matriz de ganho Φ, que varia consoante o estimador definido (LS, MV, MLS, MLSA e TDC). A reconstrução pode ser definida pela expressão:

$$\widehat{\mathbf{H}} = \mathbf{P} * \boldsymbol{\Phi} * \mathbf{V_1}^{\mathbf{T}} \tag{3.11}$$

Se a reconstrução fosse feita utilizando-se 100% dos valores próprios da matriz de Scatter do sinal de entrada X, obtinha-se $\widehat{H}$ exactamente igual a H, o que seria equivalente a dizer que não existiria ruído e a variância seria nula.

Seguidamente, refere-se o método já enunciado no capítulo 2, média das anti-diagonais [9], que permite converter os dados na forma de Hankel num único vector de dados. Com este processo foi assim possível obter novamente os dados estimados sob a forma de vector, tal como eram apresentados inicialmente.

O trecho de código seguinte demonstra a forma como este processo foi implementado.

```
%Inicialização do vector de dados estimados.
vectFrame = zeros (frameSize, 1);

%Média das anti-diagonais
    for i=1:frameSize
        vectFrame (i) = mean (diag (fliplr (Z), n-i));
    end
```

Por fim, apresenta-se um esquema que ilustra todo o processo enunciado, onde é possível identificar todos os passos descritos.

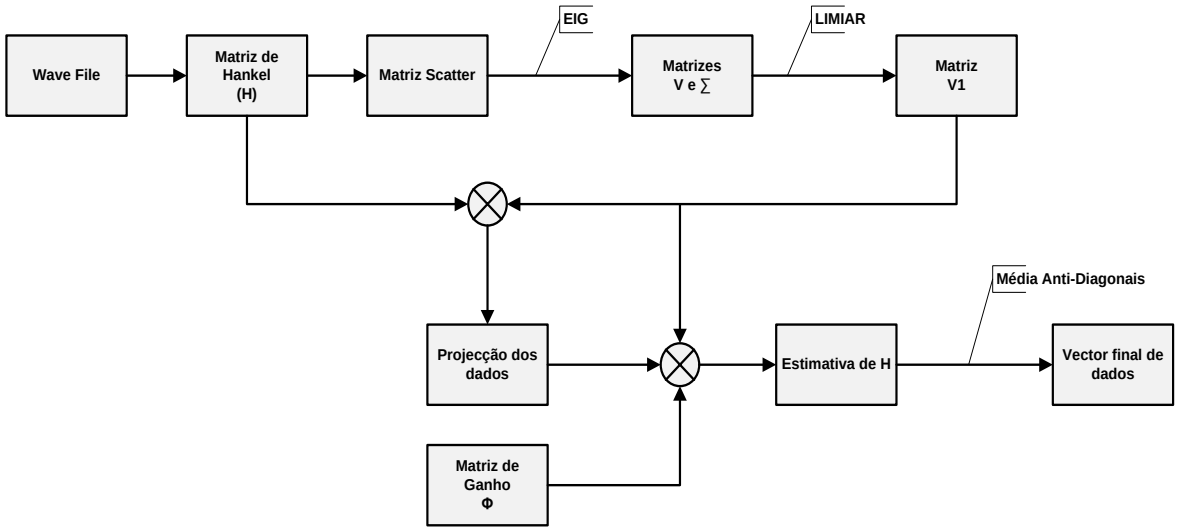

Figura 8 – Diagrama geral do método de Speech Enhancement implementado baseado em SVD

3.5.2. **Redução pelo processamento dos coeficientes MFCC e Delta**

Anteriormente foi referido que a análise do sinal era realizada frame a frame. Assumindo que cada frame do sinal a analisar é caracterizado por um conjunto de parâmetros característicos (coeficientes MFCC e Delta), para cada comando (prompt) definido nos dois cenários, podemos formar uma matriz de dados, Z. Cada linha *i* desta matriz, contém os parâmetros característicos do frame *i* do sinal de entrada. A decomposição em valores singulares de Z define-se por

$$\mathbf{Z} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathbf{T}} \tag{3.12}$$

Nesta matriz de dados podemos efectuar manipulações idênticas às descritas na secção anterior. Assumindo que parte dos valores singulares é atribuída ao ruído, pode calcular-se a matriz de dados reconstruída

$$\hat{\mathbf{Z}} = \mathbf{U_1}\boldsymbol{\Sigma_1}\boldsymbol{\Phi}\mathbf{V_1}^{\mathbf{T}} \tag{3.13}$$

Onde a matriz de ganho $\boldsymbol{\Phi}$ e a matriz $\mathbf{V_1}$ são calculadas da mesma forma descrita na secção anterior. Note-se que caso $\boldsymbol{\Phi} = \mathbf{I}$, matriz identidade, corresponde a uma aproximação idêntica à decomposição em componentes principais [4].

No método implementado, a matriz Z pode ser definida de duas formas. No primeiro caso, cada linha da matriz Z contém apenas os coeficientes MFCC (13). A reconstrução enunciada é aplicada a esta matriz, variando a matriz de ganho $\boldsymbol{\Phi}$ consoante o método pretendido *(LS, MLS, MV, TDC e MLSA)*. Posteriormente, são calculados os coeficientes delta de primeira e segunda ordem, associados aos coeficientes MFCC.

No segundo caso, cada linha da matriz Z contém os coeficientes MFCC e Delta (39). A reconstrução será aplicada a esta matriz que já contém todos os coeficientes, e tal como no caso anterior, variando a matriz de ganho $\mathbf{\Phi}$ consoante o método.

Capítulo 4 **Implementação de um reconhecedor de fala dependente do orador**

Na abordagem deste capítulo podemos destacar a descrição dos modelos HMM [6] criados na implementação do reconhecedor e a implementação do mesmo, baseado na ferramenta HTK.

4.1. Modelos HMM (Hidden Markov Model)

Na presente secção são apresentados conceitos introdutórios para a compreensão dos modelos HMM.

Consideremos um sistema composto por N estados possíveis, denominado de S_1, S_2, $S_3 \ldots S_N$, num dado instante de tempo (t). Em intervalos de tempo constantes, o sistema irá transitar de estado ou permanecer no mesmo. No entanto, se dado esse instante t, sendo o estado em que se encontra o sistema representado por q_t, a descrição probabilística deste processo estocástico requer o conhecimento dos estados ocupados nos instantes passados, podendo ser descrito pela expressão:

$$\mathbf{P}\left(\mathbf{q_t = S_j} \middle| \mathbf{q_{t-1} = S_i, q_{t-2} = S_k}, \ldots\right) \quad \mathbf{1 \leq i, j, k \leq N} \tag{4.1}$$

Se o processo de Markov for de primeira ordem, então a descrição probabilística será apenas condicionada pelo estado no instante anterior. A sua representação matricial será dada por uma matriz de transacção entre estados, denominada de $\mathbf{A} = \{\mathbf{a_{ij}}\}$, independente do instante de tempo, onde cada elemento desta matriz será dado por:

$$\mathbf{a_{ij}} = \mathbf{P}\left(\mathbf{q_t = S_j} \middle| \mathbf{q_{t-1} = S_i}\right) \quad \mathbf{1 \leq i, j \leq N} \tag{4.2}$$

Como restrição, esta matriz terá de ter cada elemento maior ou igual a zero e, a soma de todos os elementos uma dada linha da matriz de transacção tem de ser igual a 1, ou seja

$$\mathbf{a_{ij} \geq 0 \ \ e \ \ \sum_{j=1}^{N} a_{ij} = 1} \tag{4.3}$$

No caso de aplicações a casos reais, o processo referido (Markov de primeira ordem), apresenta bastantes restrições devido à sua pouca eficácia de utilização. De forma a tornar o modelo mais flexível, a cada estado é associada uma função distribuição de probabilidade de observações, podendo gerar uma observação dentro de um vasto conjunto, de acordo com esta distribuição.
Assim, a mesma sequência de observação pode ser gerada com probabilidades diferentes, através de sequências diferentes de estados. Uma vez que a sequência de estados que gera uma sequência de observações não é conhecida, este modelo denomina-se de não-observável.

Finalmente, serão apresentados os vários elementos que formam um modelo de Markov (HMM). Como elementos temos:

- ***O número N de estados*** – Denota-se por $S = \{S_1, S_2, \dots, S_N\}$ e o estado num dado instante t por q_t.

- A ***distribuição de probabilidades inicial*** para cada estado $\pi = \{\pi_i\}$.

$$\boldsymbol{\pi_i = P(q_1 = S_i) \ \ 1 \leq i \leq N} \tag{4.4}$$

- A ***distribuição de probabilidades*** de transacção entre estados definida pela matriz $A = \{a_{ij}\}$, onde a_{ij} é dado pela equação 4.2.

- O ***número de M símbolos distintos observáveis*** por estado. Este conjunto denota-se por $\mathbf{V} = \{\mathbf{V_1}, \mathbf{V_2}, \dots, \mathbf{V_M}\}$.

- A distribuição de probabilidades dos símbolos observáveis para cada estado S_j, $B = \{b_j(k)\}$, onde:

$$\mathbf{b_j(k) = P(V_k \text{ no instante } t | q_t = S_j) \quad 1 \leq j \leq N \text{ e } 1 \leq k \leq M} \tag{4.5}$$

Resumidamente, o modelo poderá ser especificado através da notação:

$$\boldsymbol{\lambda = (A, B, \pi)} \tag{4.6}$$

4.2. Criação do reconhecedor baseado em HTK

Na implementação de um reconhecedor [1], no âmbito do trabalho pretendido, destacam-se três fases distintas:

Figura 9 – Fases de implementação de um reconhecedor baseado em HTK

4.2.1. **Preparação dos dados**

Para a criação do reconhecedor serão necessários dados para treino e teste. No entanto, para a obtenção destes será necessário percorrer várias etapas, tais como:

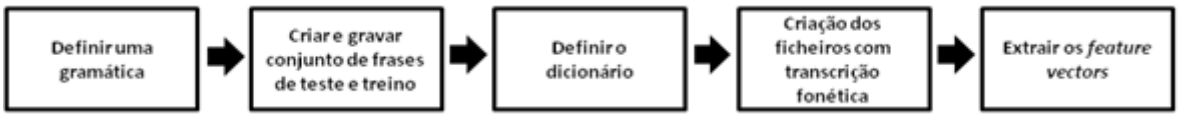

Figura 10 – Etapas para obtenção dos dados de treino e teste

Primeiramente, antes de definir uma gramática, foi necessário definir os cenários de aplicação do reconhecedor. Como já referido na secção 3.2, definiram-se dois cenários: ***controlo de cadeira de rodas*** e ***controlo de sala de cinema em casa***. No segundo cenário, as tarefas a realizar são mais complexas, permitindo termo de comparação entre os dois cenários, quer no desempenho do reconhecedor, quer do método de Speech Enhancement implementado.

- ***Definir uma gramática***

A definição da gramática [15] é uma etapa independente entre os cenários, pois as tarefas a realizar são bastante distintas. No primeiro cenário, podemos destacar tarefas como ***andar frente, parar, virar direita, direita, esquerda, virar esquerda***. No segundo, apresentam-se tarefas como ***ligar/desligar projector, subir/baixar estore, subir/baixar volume, subir/baixar tela, ligar/desligar colunas, ligar/desligar DVD, ligar/desligar luzes.***

Na figura 11, podemos observar o conteúdo do ficheiro de gramática, para o segundo cenário. É importante referir que as **barras verticais** representam casos em que **existe mais do que uma possibilidade de acção** e, as palavras delimitadas por **parêntesis recto definem palavras opcionais.**

Esta gramática é apresentada em alto nível de representação. No entanto, o HTK utiliza a gramática recorrendo a um baixo nível. Assim, é usada uma rede onde estão representadas todas as palavras, bem como a forma como estas se ligam entre si. Este tipo de notação é denominado de ***HTK Standard Lattice Format (SLF).*** Com o auxílio da ferramenta ***HParse***, disponibilizada pelo HTK, podemos criar esta mesma rede.

```
$estado = LIGAR | DESLIGAR;
$estado2 = BAIXAR | SUBIR;
$estado3 = CIMA | BAIXO | SELECCIONAR;

$objectos_a_ligar_desligar = LUZES | COLUNAS ;
$objectos_a_ligar_desligar2 = DVD | PROJECTOR ;


$objectos_baixar_subir = TELA;
$objectos_baixar_subir2 = VOLUME | ESTORE;

(
      SENT-START

      ( $estado [AS] $objectos_a_ligar_desligar  )  |
      ( $estado [O] $objectos_a_ligar_desligar2  )  |
      ( $estado2 [A] $objectos_baixar_subir )  |
      ( $estado2 [O] $objectos_baixar_subir2)  |
      ( MENU $estado3)

      SENT-END
 )
```

Figura 11 – Gramática (Cenário 2)

Para realizar a tarefa pretendida deve ser executado o comando

HParse gram wndet

onde *gram* representa o ficheiro onde se encontra definida a gramática e *wdnet* o ficheiro que contém a rede de palavras criada.

- ***Criação e gravação de um conjunto de frases de treino e teste***

Os conjuntos de frases para treino e teste podem ser obtidos com o auxílio da ferramenta ***HSGen***. Tal como o nome indica, o primeiro conjunto permite treinar os modelos HMM que vão sendo criados, automaticamente, à medida que a implementação do reconhecedor vai avançando. O segundo conjunto, permite testar o desempenho do reconhecedor após todos os modelos HMM estarem definidos. Para tal, é também necessário o uso do ficheiro

criado anteriormente que contém a rede de palavras e, um ficheiro denominado ***dicionario,*** onde se encontram palavras do dicionário genérico para o Português.

Em ambos os cenários o processo de criação destes conjuntos é igual, sendo definido por

HSGen –l –n <variavel> wdnet dicionario > TrainPrompts.txt

HSGen –l –n <variavel2> wdnet dicionario > TestPrompts.txt

onde os ficheiros *TrainPrompts.txt* e *TestPrompts.txt* representam o conjunto de frases de treino e teste, respectivamente. As opções *variavel* e *variavel2* definem o número de frases de treino e teste a criar, pretendidas pelo utilizador.

Após a criação dos conjuntos de frases de treino e teste é necessário proceder à sua gravação. Para isso recorremos à aplicação da ferramenta HTK denominada de ***HSLab***.
Para automatizar a gravação deste conjunto de frases é necessário construir um script em Perl, que execute os comandos apresentados de seguida:

HSLab train
copy train_0 S$num.wav

onde *train_0* é o ficheiro gerado pelo comando ***HSLab train*** e *$num* representa o número da frase à qual corresponde a gravação realizada, dentro do conjunto de frases de treino. Como resultado final deverão existir ficheiros áudio desde ***S001.wav até S<variavel>.wav***, correspondendo a todo o conjunto de frases criado anteriormente.

Todo o processo anterior deverá ser repetido, mas agora para o conjunto de frases de teste. O script Perl deverá agora conter a sequência de comandos apresentada de seguida.

HSLab test
copy test_0 T$num.wav

Tal como para o conjunto de treino, no final da gravação de todas as frases do conjunto de teste, deverão existir ficheiros áudio compreendidos desde ***T001.wav até T<variavel2>.wav***.

- ***Definir um dicionário***

Para a construção de um dicionário a ser utilizado pelo reconhecedor, é necessário primeiramente identificar todas as palavras presentes nas frases que se pretende reconhecer. Este dicionário [15] deverá conter a representação fonética de todas as palavras que constituem estas frases. Para obter esta lista de palavras é necessário extraí-las do conjunto de frases de treino. Assim, poderá ser usado o script fornecido pela ferramenta HTK, denominado *prompts2wlist.pl*, criado em Perl.
Como parâmetros de entrada, temos o ficheiro que contém o conjunto de frases de treino (*TrainPrompts.txt*) e, como parâmetro de saída, o ficheiro *wlist*, que contém todas as palavras utilizadas nas frases do conjunto de treino. O comando a executar deverá ser:

perl prompts2wlist.pl TrainPrompts.txt wlist

Utilizando o ficheiro *wlist* e um dicionário genérico é possível, com recurso à aplicação da ferramenta HTK denominada de ***HDMan***, criar o dicionário, contendo a pronúncia das palavras utilizadas pelo reconhecedor. Esta aplicação deverá ser usada da seguinte forma:

HDMan –m –w wlist –n monophones1 –l dlog dict dicionario

Com a execução deste comando, a opção –l permite a criação de um ficheiro *dlog* com a informação estatística sobre o dicionário e, a opção –n permite criar uma lista com todos os fonemas utilizados pelo dicionário *dict.* Esta lista de fonemas é guardada num ficheiro denominado *monophones1* (figura 13), onde se pode verificar que se encontra representado o modelo de silêncio *sil*.

No ficheiro *dict*, apresentado na figura 12, podemos verificar que este apresenta a seguinte estrutura:

WORD [outsyn] p1 p2 p3 ….

Significando que a palavra (WORD) é pronunciada com recurso à sequência de fonemas p1 p2 p3 etc. O conteúdo representado dentro de parêntesis indica o *output* do reconhecedor sempre que a palavra é reconhecida.

```
#!MLF!#
"S001.lab"
PARAR
.
"S002.lab"
PARAR
.
"S003.lab"
DIREITA
.
"S004.lab"
ESQUERDA
.
"S005.lab"
PARAR
.
"S006.lab"
ANDAR
FRENTE
.
"S007.lab"
PARAR
.
"S008.lab"
PARAR
```

Figura 14 – TrainWords.mlf

```
ANDAR           an d a r sp
DIREITA         d i r axj t ax sp
ESQUERDA        S k e r d ax sp
FRENTE          f r en t sp
PARA            p a r ax sp
PARAR           p ax r a r sp
VIRAR           v i r a r sp
```

Figura 12 – dict (cenário 1)

```
an
d
a
r
sp
i
axj
t
ax
S
k
e
f
en
p
v
sil
```

Figura 13 – monophones1 (cenário 1)

Apesar de as imagens apresentadas estarem associadas ao primeiro cenário, o processo de obtenção dos ficheiros referidos, é igual para o caso do segundo cenário. Na restante definição dos passos a realizar para a criação do reconhecedor, será referenciado apenas o primeiro cenário, pois para o segundo cenário apenas varia o conteúdo dos ficheiros.

- ***Criação dos ficheiros com a transcrição fonética***

Para que todos os dados que foram acabados de criar tenham utilidade, é necessário proceder à criação das respectivas transcrições fonéticas. Primeiramente, vamos criar um

Master Label File (MLF) com as transcrições ao nível da palavra, para cada um dos ficheiros áudio. Para criar este ficheiro de uma forma automática utilizou-se um script disponibilizado pelo HTK denominado ***prompts2mlf.pl***.

O comando a executar é:

perl prompts2mlf.pl TrainWords.mlf TrainPrompts.txt promptsFormS

onde os parâmetros de entrada são o ficheiro com as frases de treino e um ficheiro denominado *promptsFormS*, que contém o nome de cada ficheiro áudio criado, ou seja, de S001 até S500. Como parâmetro de saída, temos o ficheiro *TrainWords.mlf,* que contém as transcrições ao nível da palavra, podendo parte do seu conteúdo ser observado na figura 14.

Criado o ficheiro com a transcrição fonética ao nível da palavra, procedemos à criação do ficheiro com as transcrições fonéticas. Este ficheiro pode ser criado automaticamente, executando o seguinte comando:

HLEd –d dict –i phones0.mlf mkphones0.led TrainWords.mlf

```
#!MLF!#
"S001.lab"
sil
p
ax
r
a
r
sil
.
"S002.lab"
sil
p
ax
r
a
r
sil
.
```

Figura 15 – phones0.mlf

Temos, como parâmetros de entrada, o ficheiro ***dict*** e ***TrainWords.mlf***, ambos criados anteriormente. O ficheiro ***mkphones0.led*** contém as configurações para gerar a transcrição fonética e o parâmetro de saída é o ficheiro ***phones0.mlf***, que contém as transcrições fonéticas (figura 15).

- ***Extrair os feature vectors***

Este é o último passo a realizar na tarefa de preparação dos dados. Consiste na extracção de vectores com as características mais relevantes do sinal. Nesta parte, a extracção destes pode ser realizada de duas formas distintas, Matlab ou ferramenta HCopy do HTK.

Com o script Matlab criado, cada ficheiro áudio de uma dada directoria irá ser analisado em frames de 400 amostras. No final da execução de cada um destes ficheiros áudio, é produzida uma matriz que contém 39 colunas que correspondem aos coeficientes a calcular (coeficientes MFCC (13) e os coeficientes delta respectivos (13 velocidade e 13 aceleração)). Finalmente, por cada ficheiro áudio é criado um ficheiro mfc, cujo formato é legível pela reconhecedor HTK.

Todo este processo podia ser realizado de maneira semelhante, através do recurso à aplicação **HCopy** do HTK.

A execução desta tarefa pode ser feita recorrendo ao comando:

HCopy –T 1 –C config –S codetr.scp

A opção *–C config* indica que os parâmetros necessários para a criação dos *feature vectors* se encontram no ficheiro de configuração *config* e, *–S codetr.scp* indica que o ficheiro *codetr.scp* contém uma lista com todos ficheiros de dados e os correspondentes ficheiros de saída.

4.2.2. Criação dos modelos monofones

A fase seguinte remonta à criação de modelos monofones bem treinados. Assim, será necessário realizar as seguintes tarefas: inicialização dos modelos, ajuste dos modelos de silêncio, introdução do modelo para pausas curtas e realinhamento dos dados. Ao longo

destas tarefas, os modelos criados vão ter de ser reajustados, sendo necessária a sua re-estimação.

4.2.2.1. Inicialização dos modelos monofones

Para treinar o conjunto de HMM é necessário criar o protótipo dos modelos. Este protótipo utiliza por base uma topologia ***left-right*** com 3 estados, onde os vectores de médias e variâncias têm comprimento igual a 39, onde 13 são coeficientes MFCC, 13 coeficientes delta velocidade e 13 coeficientes delta aceleração.

Para iniciar este mesmo protótipo, é necessário substituir o valor zero dos vectores de médias e o valor um dos vectores de variâncias pelos valores globais de média e variância. Para tal, é necessário recorrer à ferramenta ***HCompV*** do HTK, e o comando a executar é o seguinte:

HCompV –f 0.01 –m –S Train.scp –M hmm0 proto

onde o ficheiro ***Train.scp*** contém uma lista com a localização dos ficheiros que contêm os *feature vectors* e, o ficheiro ***proto*** contém o protótipo necessário para a inicialização dos modelos posteriormente. Quanto às opções, ***-f*** permite que seja criado um ficheiro denominado ***vfloors***, que contém um vector igual a 0.01 vezes a variância global. A opção ***–m*** indica que as médias também são calculadas.

Posteriormente, é indispensável referir que será necessário um ***Master Macro File (MMF)***, denominado de ***hmmdefs.*** Este mesmo ficheiro deverá conter uma cópia do protótipo criado anteriormente **(hmm0/proto)**, para cada fonema presente no ficheiro ***monophones0.***

Agora que os modelos foram inicializados, é necessário proceder à sua re-estimação, de maneira a obter os modelos seguintes. A ferramenta **HERest** disponibilizada pelo HTK permite realizar esta tarefa de forma automática, da seguinte forma:

HERest -C config -I phones0.mlf -t 250.0 150.0 1000.0 -S Train.scp -H hmm0/macros –H hmm0/hmmdefs -M hmm1 monophones0

Esta ferramenta carrega os modelos presentes em **hmm0**, procedendo-se à sua re-estimação, com recurso aos dados presentes no ficheiro ***Train.scp.*** Os novos modelos criados serão guardados em **hmm1**. Com o objectivo de refinar os modelos, procede-se á criação de dois novos modelos, **hmm2 e hmm3.** Os passos necessários à obtenção destes novos modelos são:

HERest -C config -I phones0.mlf -t 250.0 150.0 1000.0 -S Train.scp -H hmm1/macros –H hmm1/hmmdefs -M hmm2 monophones0

HERest -C config -I phones0.mlf -t 250.0 150.0 1000.0 -S Train.scp -H hmm2/macros –H hmm2/hmmdefs -M hmm3 monophones0

4.2.2.2. Ajuste dos modelos de silêncio e introdução de pausas curtas

Em todos os modelos criados anteriormente, para cada um dos fonemas foram gerados três estados, incluindo o modelo de silêncio ***sil.*** Uma vez que o modelo de silêncio existente não permite a ocorrência de estados de silêncio consecutivos, é necessário alterar o modelo de forma a permitir transições entre os estados dois e quatro, em ambos os sentidos. As alterações a efectuar no modelo de silêncio estão definidas no ficheiro ***sil.hed***, cujo conteúdo pode ser observado na figura seguinte.

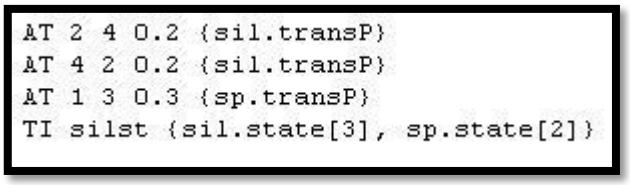

Figura 16 – sil.hed

Observando o conteúdo do ficheiro referido, podemos verificar que este obedece a uma determinada estrutura:

AT i j prob itemList(t)

Analisando um caso específico, ***AT 2 4 0.2 {sil.transP}***, esta linha tem como significado adicionar uma transição entre o estado *i* (2) e o estado *j* (4), com uma probabilidade *prob* (0.2) na matriz *t* (sil.transP).

Seguidamente, vai ser introduzido o modelo para **pequenas pausas, sp**. Para tal, deverá ser feita a cópia do ficheiro ***hmm3/hmmdefs*** e ***hmm3/macros*** para o modelo ***hmm4.*** No modelo hmm4, ***deverá ser incluído no ficheiro hmm4/hmmdefs o modelo sp*** referido anteriormente, cujos ***valores de média e variância deverão ser iguais ao 3º estado do modelo*** sil, incluído no ficheiro ***hmm3/hmmdefs***.

As alterações presentes no ficheiro ***sil.hed*** são aplicadas através da ferramenta ***HHEd*** e os novos modelos guardados em ***hmm5***, executando-se apenas o seguinte comando

HHEd -H hmm4/macros -H hmm4/hmmdefs -M hmm5 sil.hed monophones1

De seguida, serão novamente refinados os modelos. É importante referir que entre a execução dos dois comandos seguintes, deverá ser incluído o modelo de silêncio ao dicionário ***dict.***

HERest -C config -I phones0.mlf -t 250.0 150.0 1000.0 -S Train.scp -H hmm5/macros –H hmm5/hmmdefs -M hmm6 monophones1

HERest -C config -I phones0.mlf -t 250.0 150.0 1000.0 -S Train.scp -H hmm6/macros –H hmm6/hmmdefs -M hmm7 monophones1

4.2.2.3. **Realinhamento dos dados**

Até este ponto, os modelos presentes em hmm7 foram estimados tendo em consideração que existem pequenas pausas entre as palavras que constituem as frases que se pretende reconhecer. No entanto, as transcrições existentes foram geradas considerando-se que as palavras estavam encostadas umas às outras, não sendo assim consideradas as pequenas pausas entre as palavras. Como forma de rematar o problema, será necessário introduzir as pequenas pausas existentes nos modelos criados. O comando a executar para realizar esta tarefa automaticamente será:

HVite -o SWT -b silence -C config -a -H hmm7/macros -H hmm7/hmmdefs -i aligned.mlf -m –t 250.0 -y lab -I TrainWords.mlf -S Train.scp dict monophones1

```
"S005.lab"
sil
p
ax
r
a
r
sp
sil
.
"S006.lab"
sil
an
d
a
r
sp
f
r
en
t
sp
sil
.
"S007.lab"
sil
p
ax
r
a
r
sp
sil
.
```

Figura 17 – aligned.mlf

A figura 17, ilustra o conteúdo do ficheiro resultante da execução deste comando, ***aligned.mlf***, e verifica-se que existem pequenas pausas (sp), entre as palavras que constituem uma frase que se pretende reconhecer.

Tal como já foi feito anteriormente, será necessário estimar mais dois conjuntos de modelos, ***hmm8 e hmm9.*** Os comandos a executar serão:

HERest -C config -I aligned.mlf -t 250.0 150.0 1000.0 -S Train.scp -H hmm7/macros -H hmm7/hmmdefs -M hmm8 monophones1

HERest -C config -I aligned.mlf -t 250.0 150.0 1000.0 -S Train.scp -H hmm8/macros -H hmm8/hmmdefs -M hmm9 monophones1

4.2.3. Avaliação do Reconhecedor

Após a criação dos modelos utilizando os dados de treino, procede-se à obtenção de resultados relativos ao desempenho do reconhecedor [7], recorrendo-se ao conjunto de frases de teste. É necessário copiar os modelos obtidos em ***hmm9*** e guardá-los em ***hmm15*** e deverá ser criado o ficheiro ***TestWords.mlf***, como no caso do conjunto de frases de treino. A criação deste ficheiro deve ser realizada recorrendo ao comando apresentado.

perl prompts2mlf.pl TestWords.mlf TestPrompts.txt promptsFormT

O passo seguinte é a criação do ficheiro ***recout.mlf***, indispensável para avaliação dos modelos monofones. Antecipadamente, deve também ser criado o ficheiro ***Test.scp,*** que contém a localização dos feature vectors obtidos para o conjunto de dados de teste.

O comando a executar para criar o ficheiro ***recout.mlf*** é:

HVite –H hmm15/macros –H hmm15/hmmdefs –S Test.scp –i recout.mlf –w wdnet –p 0.0 –s 5.0 dict2 monophones1

Os resultados são apresentados através da execução do comando seguinte, cujo *output* pode ser "direccionado" para um ficheiro txt.

HResults –f –h –t –I TestWords.mlf monophones1 recout.mlf > ficheiroResultados.txt

4.2.4. **Validação do Reconhecedor**

Na presente secção, serão apresentados resultados do desempenho da primeira versão do reconhecedor base criado. Como referido no capítulo anterior, a função Matlab implementada para extrair os ***coeficientes mfcc*** deverá ter um desempenho similar à da ferramenta HCopy [17] do HTK. Assim, para os cenários criados, comparámos a percentagem de reconhecimento, utilizando cada uma das ferramentas anteriores. Não foi adicionado qualquer tipo de ruído nem utilizado Speech Enhancement, em nenhum dos dois conjuntos de sinais áudio, treino e teste.

Antes de obter os resultados, foi necessário definir o número de comandos *(prompts)* que formam os dois conjuntos de sinais, treino e teste, para cada um dos cenários. No primeiro cenário foram definidas ***200 prompts*** no conjunto de treino e ***100 prompts*** no conjunto de teste. No segundo cenário, utilizaram-se ***250 prompts*** para treino e ***200 prompts*** para teste. Como no segundo cenário a gramática definida é mais complexa quando comparada com o primeiro, aumentou-se o número de *promtps* de cada conjunto.

Após a criação do reconhecedor, seguindo cuidadosamente todos os passos enunciados ao longo deste capítulo para esse efeito, foram comparadas as percentagens de reconhecimento obtidas em cada cenário.

	Cenário 1	*Cenário 2*
HCopy	99.32%	87.26%
Matlab	94.52%	83.44%

Tabela 2 – Comparação de resultados - Matlab e HCopy

Observando a tabela anterior, verificamos que o desempenho da função Matlab implementada é da mesma ordem de grandeza da ferramenta HCopy, para ambos os cenários, o que permite concluir que o objectivo pretendido foi conseguido.

No capítulo seguinte serão apresentados todos os resultados obtidos nas várias situações de estudo, como a adição de ruído e utilização de Speech Enhancement nos conjuntos de treino e teste, consoante o objectivo pretendido em cada caso.

Capítulo 5 **Resultados**

No presente capítulo são apresentados os resultados relativos ao desempenho do reconhecedor de fala robusto baseado em HTK e dependente do orador, nas mais variadas condições. De modo a verificar o desempenho deste, foi adicionado ruído branco e babble aos conjuntos de sinais de áudio (treino e teste) e, aplicado o método de Speech Enhancement implementado.

O efeito destas duas componentes tem por objectivo verificar a necessidade de treinar o reconhecedor base, adicionando ruído ao conjunto de sinais de áudio de treino e, testar o desempenho do método de Speech Enhancement, no reconhecimento de todas as palavras que formam os comandos criados para cada cenário.

Como já foi referido na secção 3.2 do capítulo 3, cada comando associado a um dos dois cenários criados, pode ser composto por uma ou mais palavras. O reconhecedor de fala para cada um dos testes realizados é treinado de forma diferente, sendo utilizado um conjunto de sinais de áudio de treino para o efeito. Para avaliar o reconhecedor, utiliza-se um conjunto de sinais de áudio de teste, verificando-se a percentagem de reconhecimento deste, ou seja, para todos os comandos do conjunto de teste, a percentagem de palavras reconhecidas consoante o treino realizado.

5.1. Desempenho do reconhecedor base utilizando apenas Speech Enhancement

Nesta secção serão apresentados os resultados relativos ao efeito da utilização de Speech Enhancement. O objectivo é verificar o desempenho deste, sem ser necessário adicionar ruído a qualquer dos conjuntos, treino e teste. Posteriormente, em secções mais avançadas, serão apresentados resultados relativos à utilização de Speech Enhancement, em condições de adição de ruído aos sinais áudio.

O reconhecedor base foi treinado em **modo "clean"**, ou seja, sem adição de qualquer tipo de ruído nem utilização de Speech Enhancement. Ao conjunto de teste também não foi

adicionado ruído, apenas foi utilizado Speech Enhancement. Como já referido no capítulo 2, foi definido um valor LIMIAR, que representa a percentagem, em relação à soma total, de valores próprios a utilizar no processamento. Para efeitos de teste, o LIMIAR definido foi de 80%.

Na tabela seguinte apresentam-se os resultados para os dois cenários criados.

Método de Speech Enhancement	*Cenário 1*	*Cenário 2*
Least-Squares	91.78%	83.44%
Modified Least-Squares	91.10%	83.44%
Minimum Variance	89.04%	83.44%
Time-Domain Constraint	96.58%	84.71%
MLSA	87.67%	80.89%

Tabela 3 – Efeito de Speech Enhancement no desempenho do reconhecedor

Em ambos os cenários, o método que apresenta melhor percentagem de reconhecimento é o método ***TDC***, enquanto o menos eficaz é o ***MLSA***. Podemos ainda comparar os resultados obtidos globalmente em cada cenário e verificar que o primeiro cenário tem melhor eficácia, pois os comandos associados a este são mais simples, quando comparados com o segundo.

5.2. Efeito de adição de ruído no desempenho do reconhecedor base

A adição de ruído aos conjuntos de treino e teste está dividida em dois casos de estudo:

- Ruído Branco
- Ruído Babble

Inicialmente, começamos por retratar a alteração dos valores próprios de cada frame de um dado sinal de áudio, aquando da adição de ruído, comparando-se dois comandos diferentes utilizados pelo reconhecedor no primeiro cenário: *parar e virar direita*.

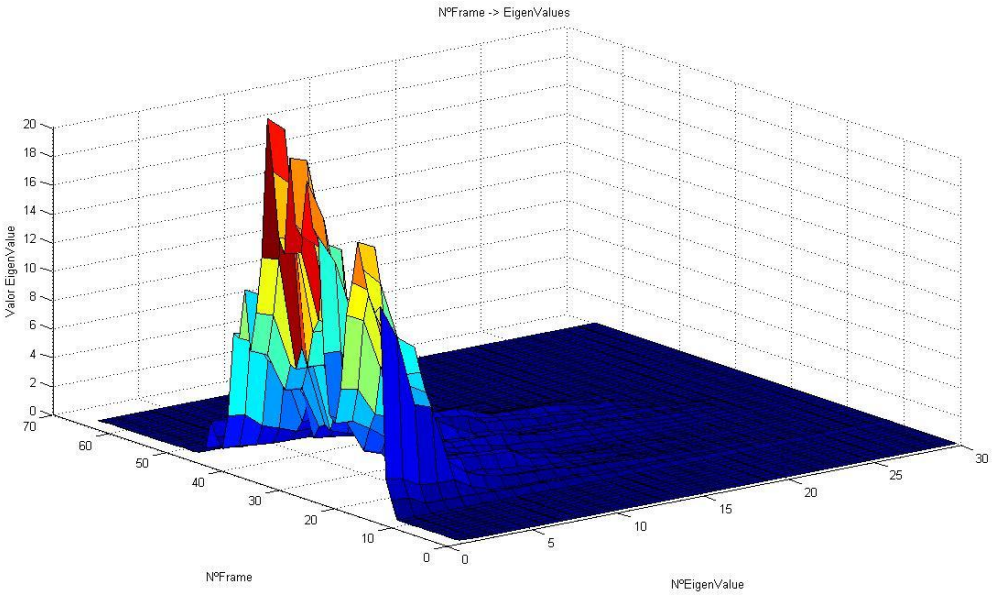

Figura 18 – Valores próprios de cada frame do comando *parar*, sem adição de ruído

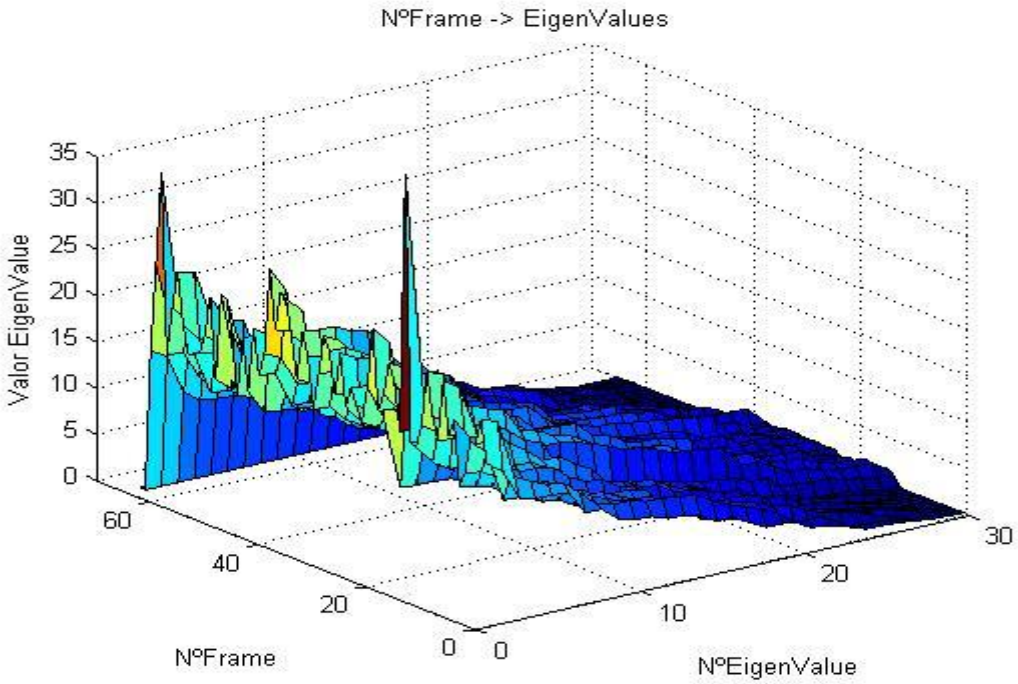

Figura 19 – Valores próprios de cada frame do comando *parar*, com adição de ruído branco (SNR=5dB)

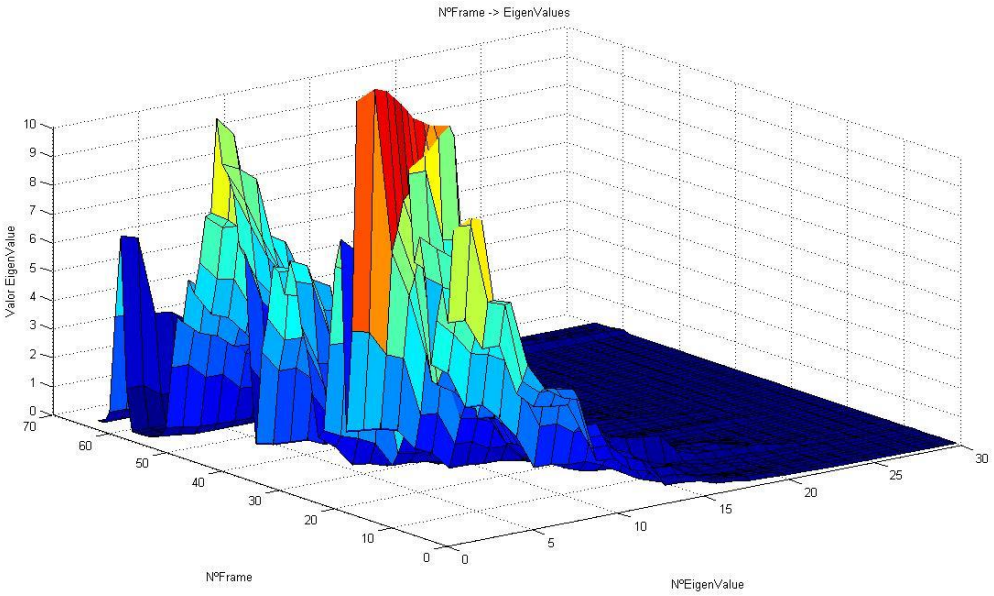

Figura 20 – Valores próprios de cada frame do comando *virar direita*, sem adição de ruído

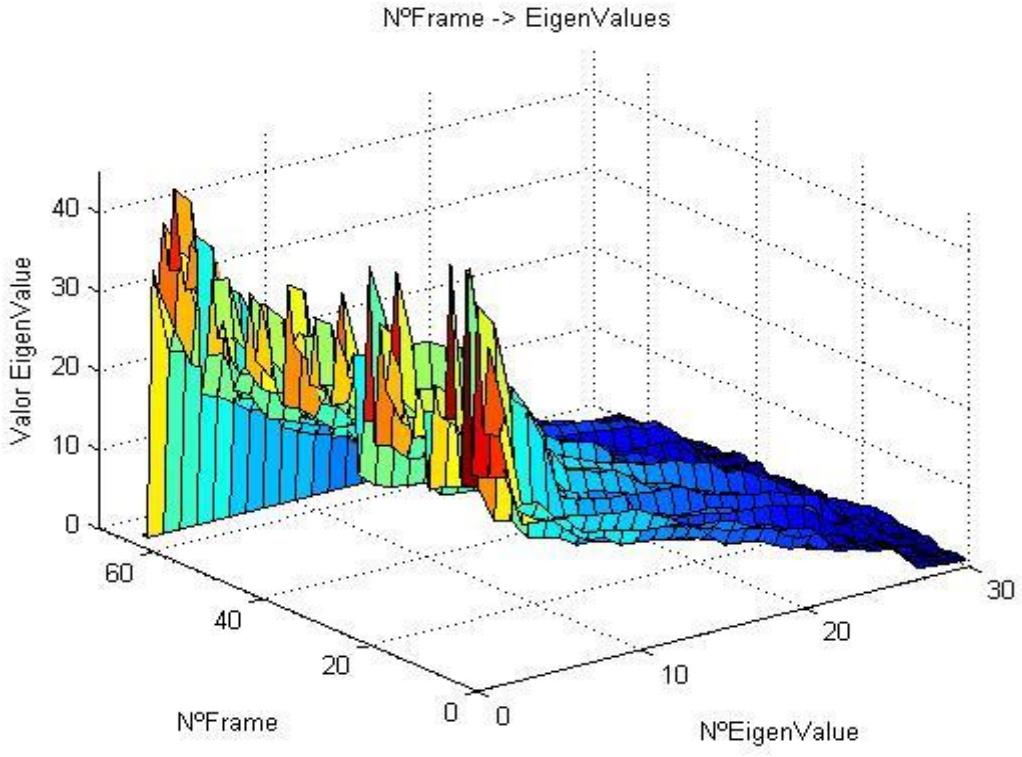

Figura 21 – Valores próprios de cada frame do comando *virar direita*, com adição de ruído branco (SNR = 5dB)

Observando-se os gráficos anteriores, para qualquer um dos comandos apresentados, verificamos que com a adição de ruído branco ao sinal os valores próprios, quando comparados com os do sinal original, estão "shiftados" por $\sigma_R{}^2$ (variância do ruído), tal como enunciado no capítulo 2. Assim, os valores próprios do sinal após a adição de ruído branco têm valor superior aos do sinal original, como é possível verificar comparando as figuras 18 e 19 e as figuras 20 e 21, que representam comandos distintos.

De seguida, são apresentados os resultados na adição dos dois tipos de ruído utilizados. Os testes realizados são equivalentes para ambos os casos. Quando for referida a adição de ruído branco ou babble ao longo deste capítulo, deve considerar-se que as características de ambos permanecem inalteráveis.

5.2.1. **Adição de Ruído Branco**

Em todos os testes aos quais foi adicionado ruído branco, com vários níveis de SNR (0,5,10 e 15 dB), recorreu-se ao comando *awgn* do Matlab [14], referenciado no capítulo 3, secção 3.4. Com a utilização deste comando, o valor do ruído a adicionar não foi sempre o mesmo, por isso realizámos dez experiências nos casos em que foi adicionado ruído branco ao conjunto de teste. No entanto, quando para além do ruído também era utilizado Speech Enhancement neste conjunto, apenas se realizou uma única experiência, pois o processamento de dados associado é muito demorado, sendo impossível realizar dez experiências para cada um dos métodos e, consequentemente, para cada nível de SNR a este associado.

5.2.1.1. **Efeito da adição de ruído branco entre treino e teste**

Nesta fase de resultados, o reconhecedor base foi treinado de duas formas distintas. No caso A, o reconhecedor base foi treinado em modo "clean", sendo adicionado ruído branco ao conjunto de teste. De seguida, no caso B, a ambos os conjuntos, treino e teste, foi adicionado ruído branco. Com a comparação entre estes dois casos, pretendemos ter apenas

um factor em estudo, o ***ruído***, verificando-se a necessidade de treinar, ou não, o reconhecedor base com a adição deste ao conjunto de treino.

Nos gráficos seguintes, podemos observar os resultados obtidos para os dois casos de estudo, nos dois cenários existentes, e comparar a percentagem de reconhecimento do reconhecedor base em cada um deles.

Cenário 1- Controlo de cadeira de rodas

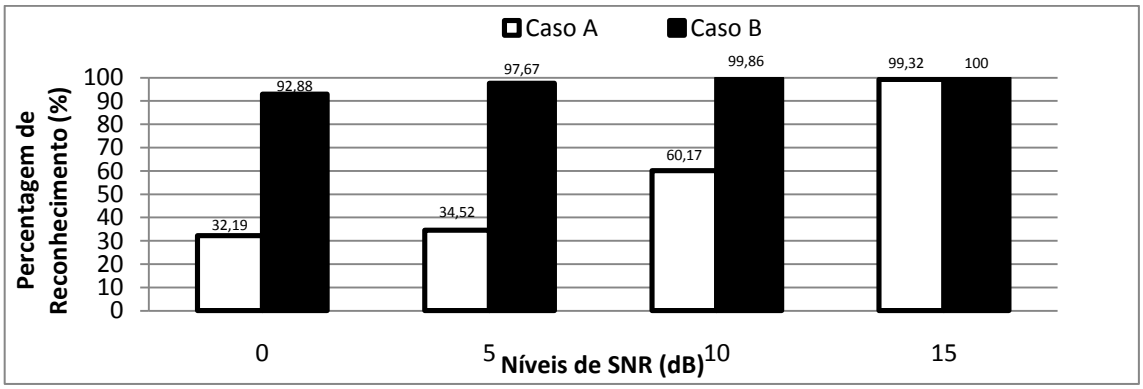

Figura 22 – Efeito da adição de ruído branco entre treino e teste (Cenário 1)

Cenário 2 – Controlo de sala de cinema em casa

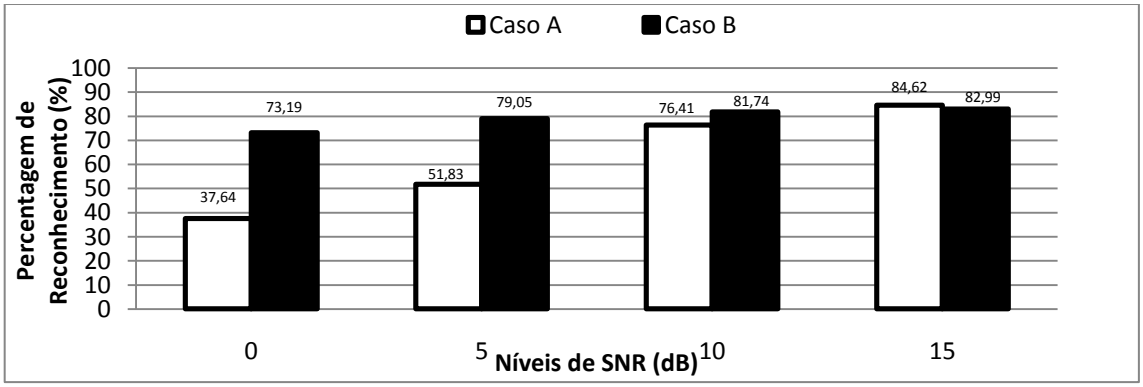

Figura 23 – Efeito da adição de ruído branco entre treino e teste (Cenário 2)

Observando os gráficos, verificamos que globalmente o caso B é melhor que o caso A, para os vários níveis de SNR. Concluímos então que existe necessidade de treinar o reconhecedor adicionando ruído ao conjunto de treino, salientando-se este facto para níveis de SNR mais baixos, 0 e 5 dB, onde a percentagem de reconhecimento entre os dois casos é mais discrepante.

5.2.1.2. **Efeito da adição de ruído branco entre treino e teste, aplicando Speech Enhancement aos sinais áudio de teste**

Na presente secção existem dois casos de estudo distintos. No caso A, em semelhança à secção anterior, o reconhecedor base foi treinado em modo "clean", enquanto ao conjunto de teste foi adicionado ruído branco e aplicado Speech Enhancement. Por outro lado, no caso B, o conjunto de teste permaneceu inalterado em relação ao caso A, tendo o reconhecedor base sido treinado com um conjunto de treino ao qual foi adicionado ruído branco. Com estes casos, pretendemos verificar se o facto de ter sido utilizado Speech Enhancement no conjunto de teste beneficia a percentagem de reconhecimento.

Nos gráficos seguintes são apresentados os resultados para os dois casos, nos dois cenários existentes, consoante o método de Speech Enhancement utilizado: LS, MLS, MLSA, TDC e MV

Cenário 1 – Controlo de cadeira de roda

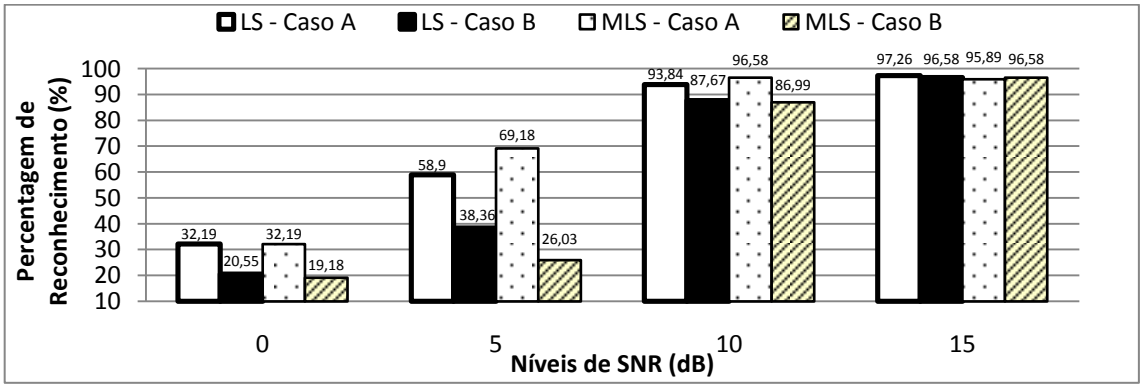

Figura 24 – Efeito da adição de ruído branco entre treino e teste, aplicando SE aos sinais áudio de teste (método LS e MLS – Cenário 1)

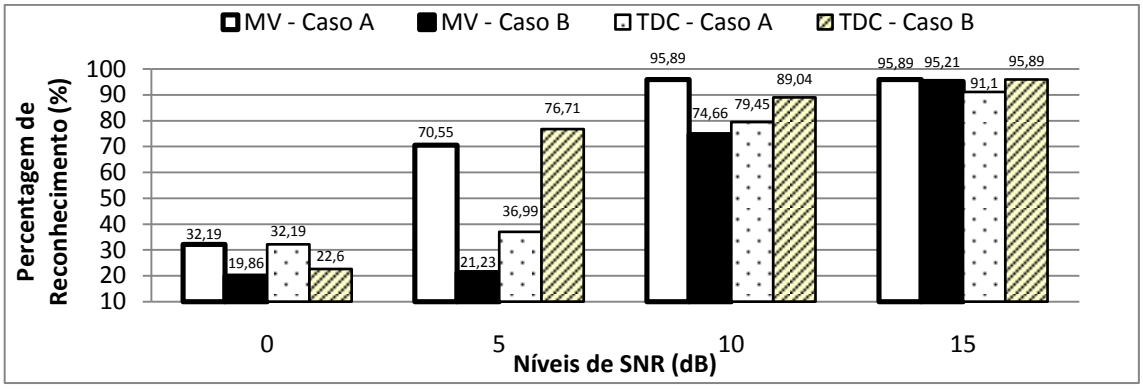

Figura 25 – Efeito da adição de ruído branco entre treino e teste, aplicando SE aos sinais áudio de teste (método MV e TDC – Cenário 1)

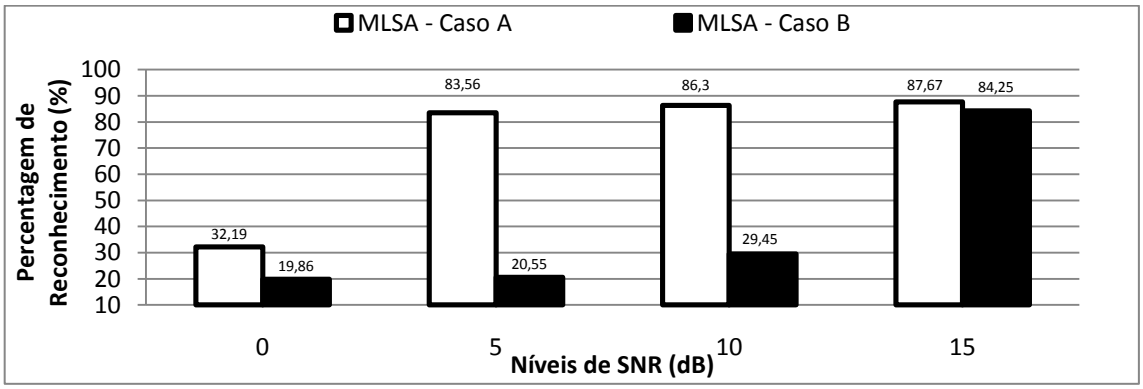

Figura 26 – Efeito da adição de ruído branco entre treino e teste, aplicando SE aos sinais áudio de teste (método MLSA – Cenário 1)

Através da análise gráfica, podemos concluir que aplicando Speech Enhancement ao conjunto de teste não existe necessidade de treinar o reconhecedor adicionando ruído, pois globalmente o caso A apresenta melhor desempenho. No entanto, esta diferença é mais saliente para níveis de SNR mais baixos. Contrariamente à secção anterior, onde a adição de ruído ao conjunto de treino era um factor determinante para a melhoria do desempenho do reconhecedor, a utilização de Speech Enhancement no teste permite melhorar a eficácia, sem haver necessidade de treinar o reconhecedor com ruído.

Cenário 2 – Controlo de sala de cinema em casa

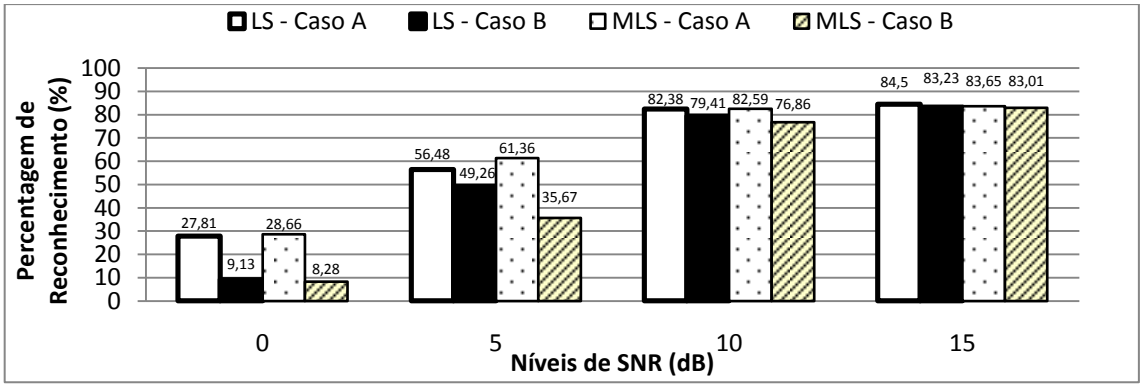

Figura 27 – Efeito da adição de ruído branco entre treino e teste, aplicando SE aos sinais áudio de teste (método LS e MLS – Cenário 2)

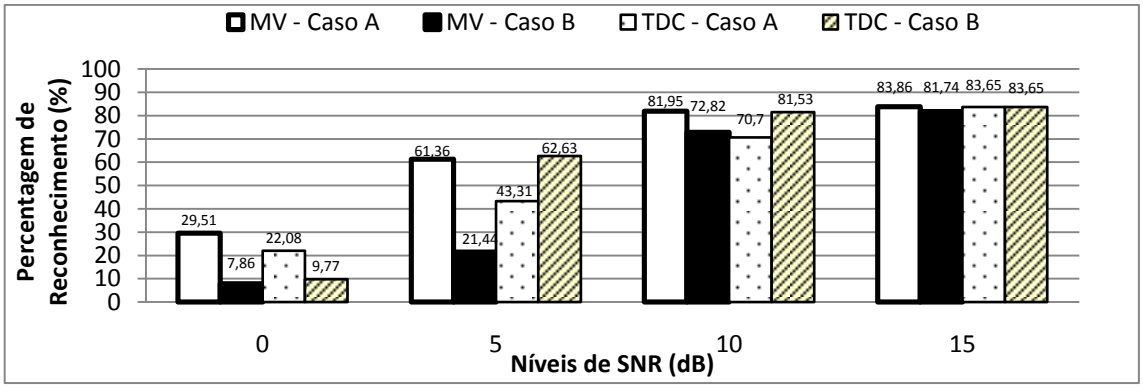

Figura 28 – Efeito da adição de ruído branco entre treino e teste, aplicando SE aos sinais áudio de teste (método MV e TDC – Cenário 2)

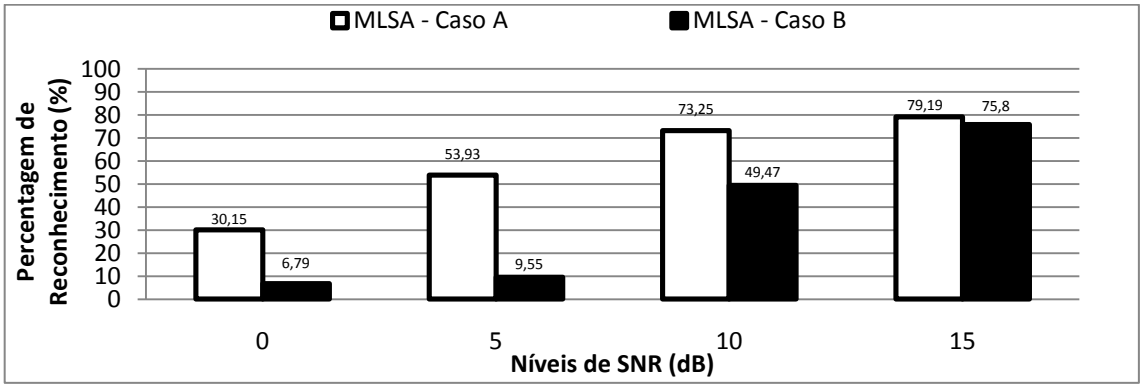

Figura 29 – Efeito da adição de ruído branco entre treino e teste, aplicando SE aos sinais áudio de teste (método MLSA – Cenário 2)

Tal como no primeiro cenário, a utilização de Speech Enhancement no conjunto de teste permite que o factor ruído não seja determinante, não existindo necessidade de treinar o reconhecedor adicionando este último aos sinais áudio. Apenas para os níveis mais baixos de SNR a diferença de percentagem de reconhecimento entre os dois casos é mais notória, enquanto para níveis superiores é equivalente.

5.2.1.3. **Efeito de utilização de Speech Enhancement no desempenho do reconhecedor base**

Na presente secção, será feita a comparação entre os resultados obtidos com e sem a utilização de Speech Enhancement nos sinais áudio do conjunto de teste, ou seja, comparamos os resultados das duas secções anteriores para os dois cenários e casos (A e B). O objectivo principal é verificar se a utilização de Speech Enhancement no conjunto de teste é benéfica para o desempenho do reconhecedor, contrariamente aos casos anteriores em que o factor de estudo era o ruído.

Apenas foram comparados resultados relativos aos dois níveis de SNR mais baixos, 0 e 5 dB, visto que para níveis superiores (10 e 15 dB) a diferença entre os casos A e B não é significativa.

A primeira comparação, ilustrada com os gráficos seguintes, compara o *Teste 5.2.1.1 – Caso A,* conjunto teste com adição de ruído e sem utilização de Speech Enhancement e reconhecedor treinado em modo "clean", com o *Teste 5.2.1.2 – Caso A,* igual ao anterior mas com utilização de Speech Enhancement no conjunto de teste.

Cenário 1 – Controlo de cadeira de rodas

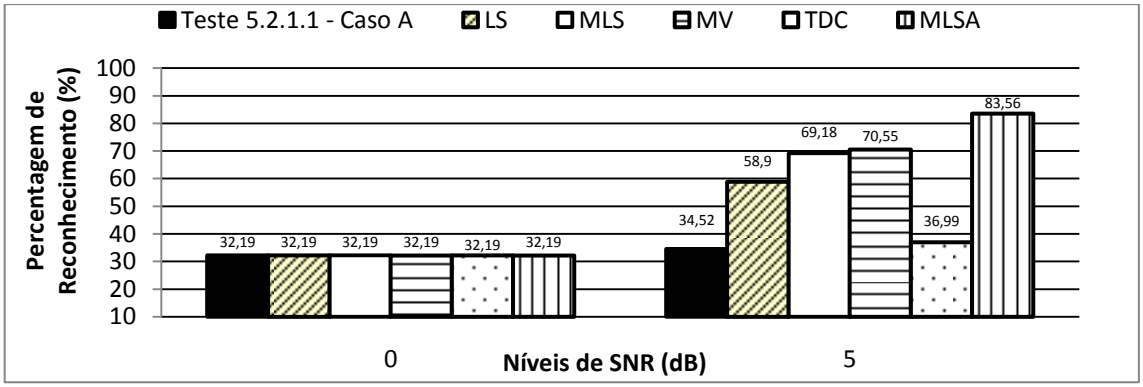

Figura 30 – Efeito de Speech Enhancement no desempenho do reconhecedor base (Cenário 1 – Caso A)

Cenário 2 – Controlo de sala de cinema em casa

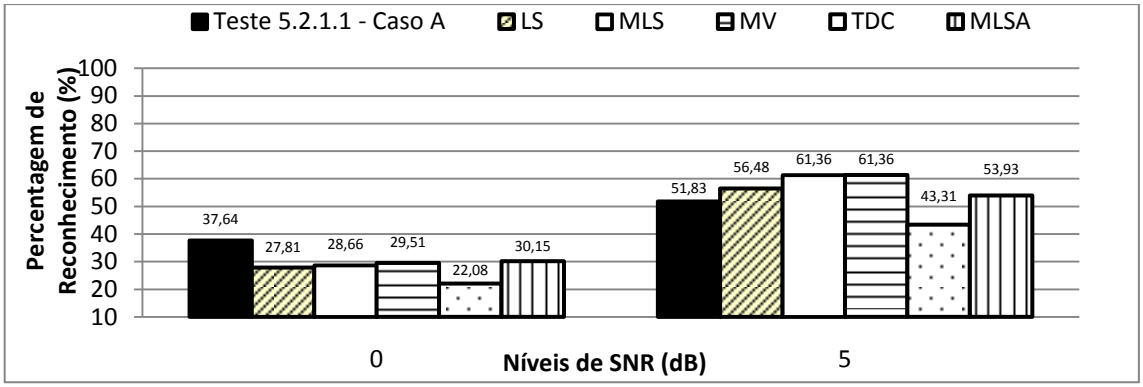

Figura 31 – Efeito de Speech Enhancement no desempenho do reconhecedor base (Cenário 2 – Caso A)

Observando os gráficos anteriores concluímos que, no primeiro cenário, para um SNR de 0dB, a percentagem de reconhecimento não é alterada com a introdução de Speech Enhancement no conjunto de teste. No entanto, para 5dB, verificamos que a utilização de Speech Enhancement é benéfica, melhorando o desempenho para todos os métodos implementados, destacando-se o ***MLSA***. No que respeita ao segundo cenário, para o nível de 0dB, a utilização de Speech Enhancement não torna o reconhecedor mais eficaz. Por outro lado, com 5dB de SNR, apenas o método ***TDC*** não produz melhores resultados.

De seguida, são apresentados os resultados comparativos para o caso B. Neste caso, no *Teste 5.2.1.1* não foi utilizado Speech Enhancement em nenhum dos conjuntos, sendo adicionado ruído a ambos. Por outro lado, no *Teste 5.2.1.2* foi adicionado ruído ao

conjunto de treino e teste, utilizando-se Speech Enhancement no último. As figuras seguintes ilustram a comparação de resultados entre os dois Testes, no caso B.

Cenário 1 – Controlo de cadeira de rodas

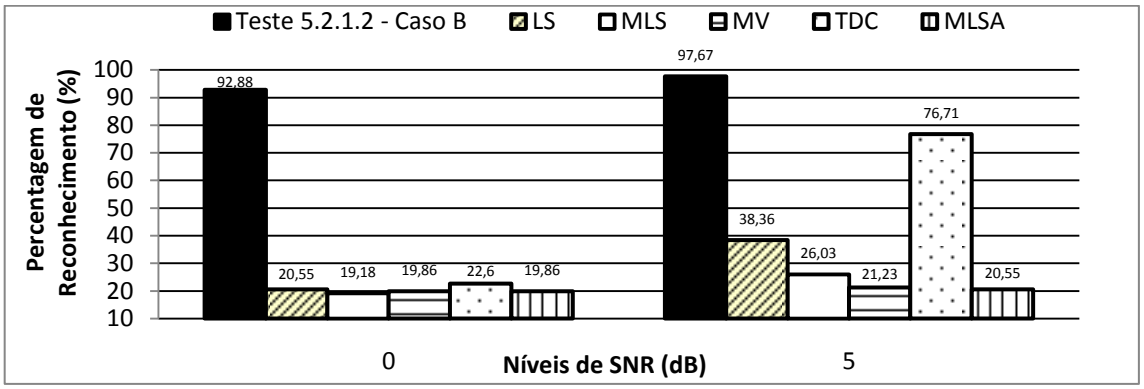

Figura 32 – Efeito de Speech Enhancement no desempenho do reconhecedor base (Cenário 1 – Caso B)

Cenário 2 – Controlo de sala de cinema em casa

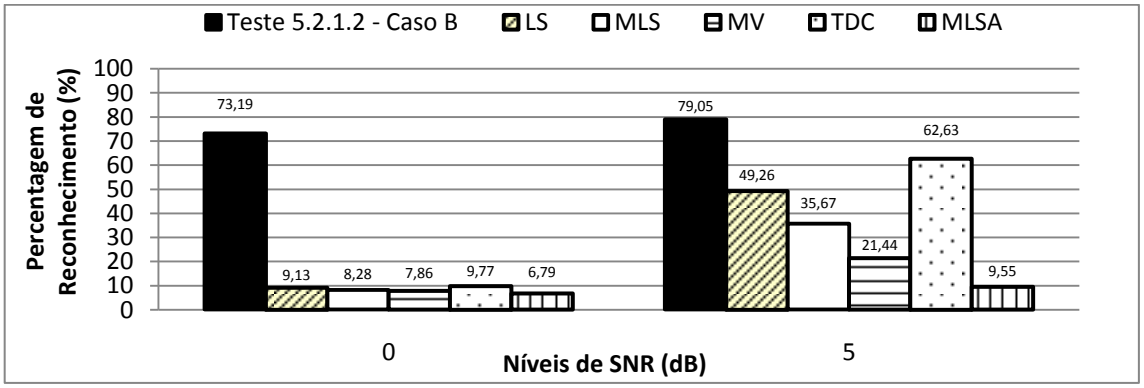

Figura 33 – Efeito de Speech Enhancement no desempenho do reconhecedor base (Cenário 2 – Caso B)

Pela análise gráfica, podemos verificar que nenhum método de Speech Enhancement utilizado produz resultados muito satisfatórios, quando comparados com o caso em que este não é usado. No entanto, quer no primeiro quer no segundo cenário, para um nível de SNR de 5dB, o método ***TDC*** é o que apresenta resultados mais próximos do caso em que não é utilizado Speech Enhancement nos sinais do conjunto de teste.

5.2.2. **Adição de Ruído Babble**

Na secção seguinte, serão apresentados os resultados relacionados com a adição de Ruído Babble. Deve salientar-se que o ruído adicionado foi sempre o mesmo, não existindo alteração nos valores de ruído adicionado, como acontecia no caso do ruído branco. Não foi possível também adicionar vários tipos de ruído babble, pois a base de dados utilizada apenas englobava um ficheiro áudio, variando apenas o nível de SNR.

Como os testes realizados para este tipo de ruído são idênticos aos do caso do ruído branco, apenas serão referenciadas as metodologias aplicadas (adição de ruído e/ou método de Speech Enhancement) aos dois conjuntos de sinais de áudio, treino e teste.

5.2.2.1. **Efeito da adição de ruído babble entre treino e teste**

Este teste é idêntico ao realizado para o caso do ruído branco. No caso A, o reconhecedor base foi treinado em modo "clean" e ao conjunto de teste foi adicionado ruído babble. Por outro lado, no caso B, foi adicionado ruído babble aos dois conjuntos de sinais áudio.

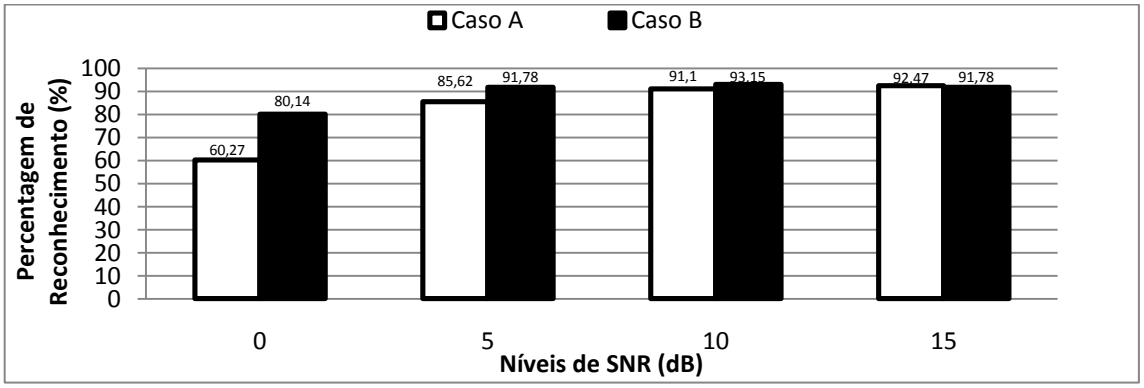

Figura 34 – Efeito da adição de ruído babble entre treino e teste (Cenário 1)

Cenário 2 – Controlo de sala de cinema em casa

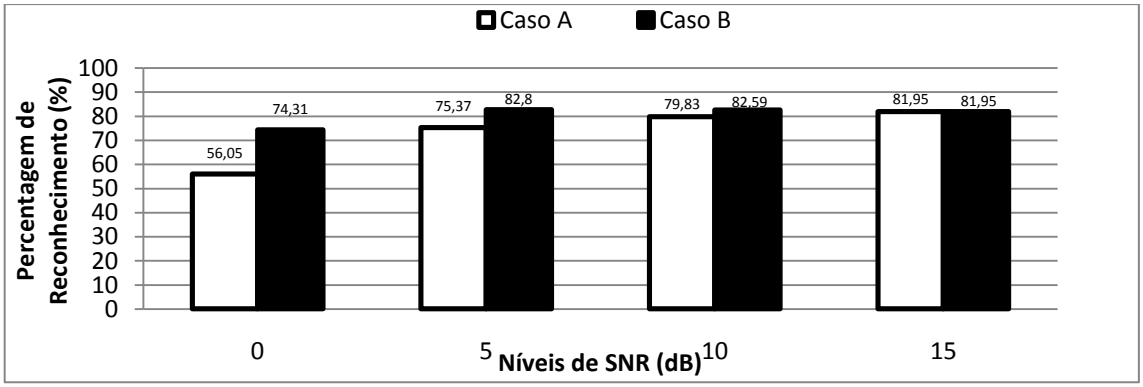

Figura 35 – Efeito da adição de ruído babble entre treino e teste (Cenário 2)

Comparando os dois casos verificamos que o caso B tem melhor desempenho que o caso A para os vários níveis de SNR e ambos os cenários. Assim, como para o ruído branco, concluímos que o factor ruído é determinante para o aumento da percentagem de reconhecimento. Comparando estes resultados com os do ruído branco, observamos que no caso A a percentagem de reconhecimento para níveis de SNR mais baixos (0 e 5 dB) é superior. Esta diferença pode estar associada ao uso do comando *awgn* do Matlab para adicionar ruído branco aos sinais, pois os valores não são sempre os mesmos. Por outro lado, o ruído babble adicionado aos sinais era sempre o mesmo, recorrendo-se à função *addNoise*.

5.2.2.2. Efeito da adição de ruído babble entre treino e teste, aplicando Speech Enhancement aos sinais de áudio e teste

O teste realizado nesta secção consiste na adição de ruído babble aos dois conjuntos de sinais áudio, treino e teste, sendo ainda aplicado ao último o método de Speech Enhancement implementado.

Nos gráficos seguintes são apresentados os resultados para os dois cenários definidos e todos os métodos implementados (LS, MLS, MV, TDC e MLSA)

Cenário 1 – Controlo de cadeira de rodas

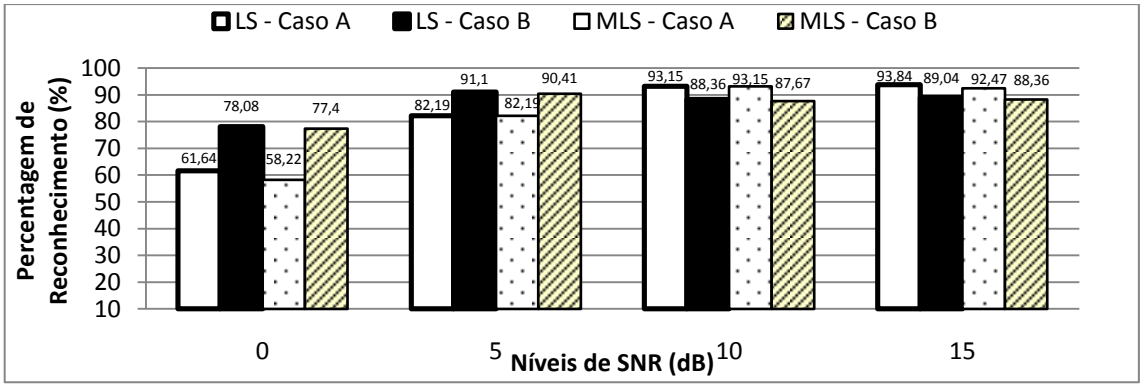

Figura 36 – Efeito da adição de ruído babble entre treino e teste, aplicando SE aos sinais áudio de teste (método LS e MLS – Cenário 1)

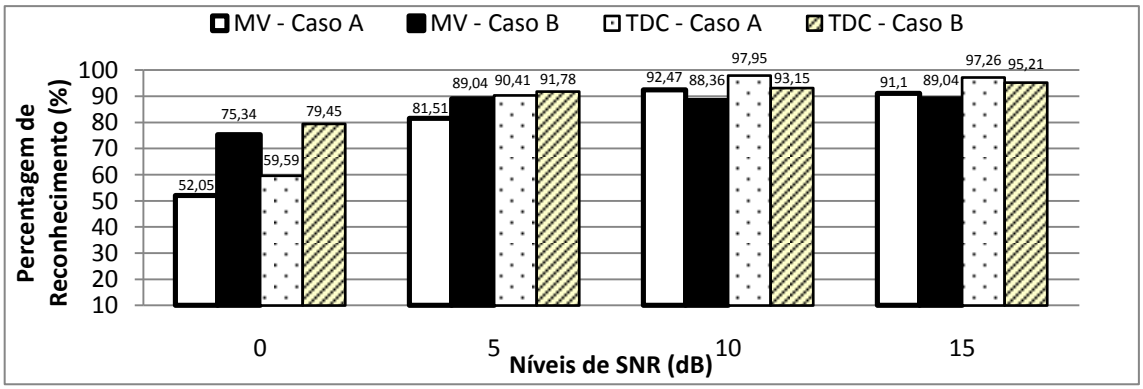

Figura 37 – Efeito da adição de ruído babble entre treino e teste, aplicando SE aos sinais áudio de teste (método MV e TDC – Cenário 1)

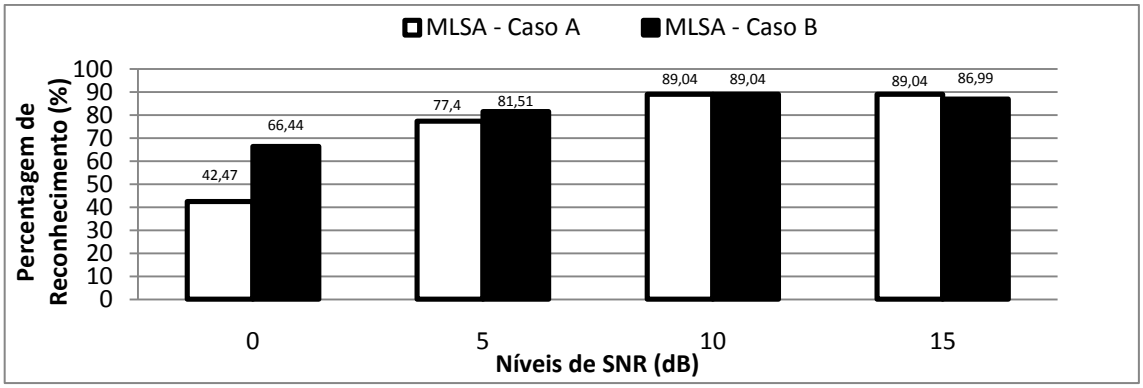

Figura 38 – Efeito da adição de ruído babble entre treino e teste, aplicando SE aos sinais áudio de teste (método MLSA – Cenário 1)

Contrariamente ao que acontecia com o ruído branco, analisando os gráficos anteriores verificamos que treinar o reconhecedor com ruído é determinante para uma melhor desempenho deste. No caso do ruído babble, a utilização de Speech Enhancement no conjunto de teste sem adicionar ruído ao treino, não é suficiente para o aumento da percentagem de reconhecimento. Tal como na secção anterior, para níveis inferiores de SNR (0 e 5 dB), a percentagem de reconhecimento adicionando ruído babble é bastante superior. Esta diferença pode estar associada ao enunciado anteriormente, ou seja, ao método de adição de ruído, comando *awgn* ou função *addNoise*, consoante é adicionado ruído branco ou babble, respectivamente.

Cenário 2 – Controlo de sala de cinema em casa

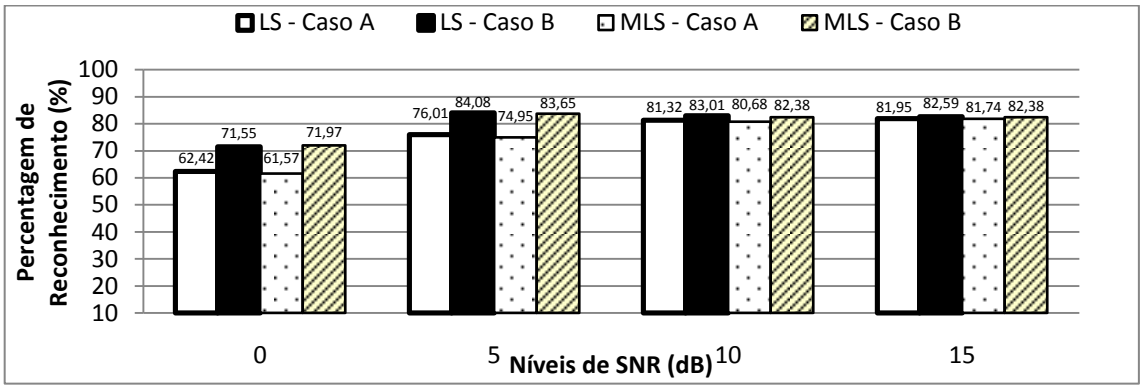

Figura 39 – Efeito da adição de ruído babble entre treino e teste, aplicando SE aos sinais áudio de teste (método LS e MLS – Cenário 2)

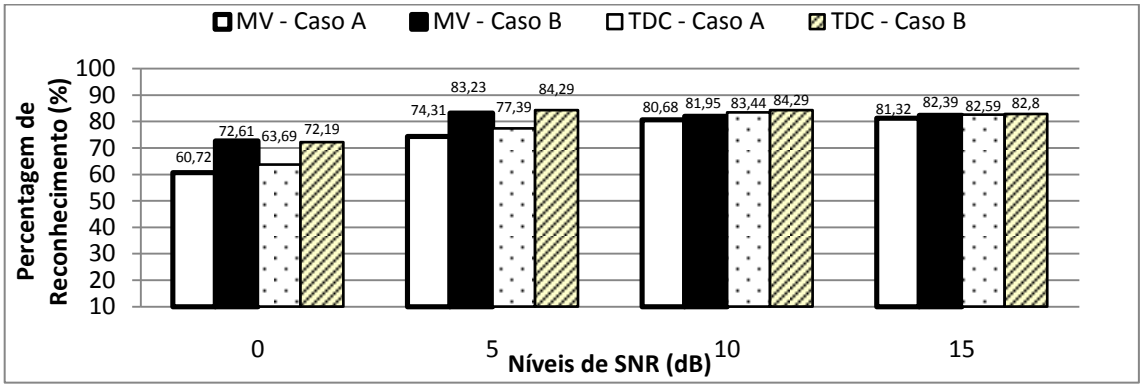

Figura 40 – Efeito da adição de ruído babble entre treino e teste, aplicando SE aos sinais áudio de teste (método MV e TDC – Cenário 2)

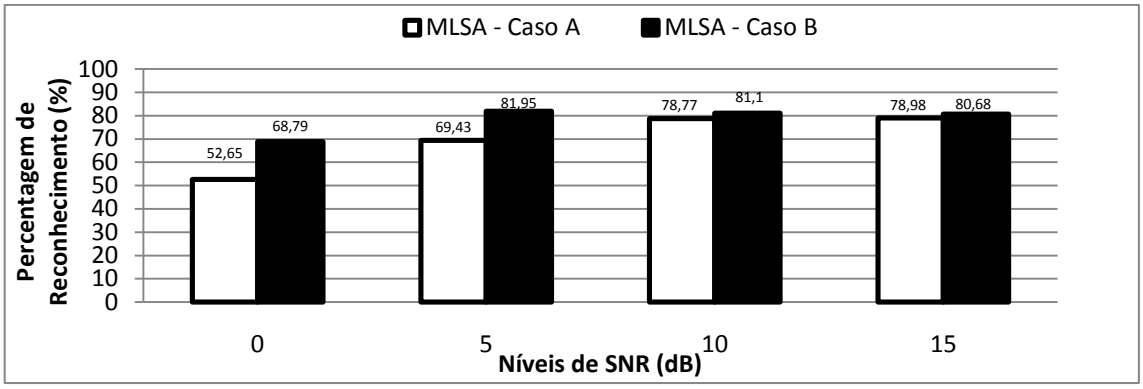

Figura 41 – Efeito da adição de ruído babble entre treino e teste, aplicando SE aos sinais áudio de teste (método MLSA – Cenário 2)

Exactamente como no primeiro cenário, a adição de ruído ao treino é fundamental para aumentar o desempenho do reconhecedor, sendo esta diferença mais evidente para níveis de SNR mais baixos. Comparando com o ruído branco, verificamos que, para níveis inferiores de SNR, a percentagem de reconhecimento é bastante superior, pelas razões já apresentadas anteriormente.

5.2.2.3. **Efeito de utilização de Speech Enhancement no desempenho do reconhecedor base**

Nesta secção será feita a comparação entre os resultados obtidos nas duas secções anteriores, verificando-se se a utilização de Speech Enhancement no teste é útil para melhorar o desempenho do reconhecedor, como sucedido no caso do ruído branco. Apenas serão analisados os dois níveis de SNR mais baixos, 0 e 5 dB, uma vez que para níveis superiores, 10 e 15 dB, os resultados obtidos são similares.

As comparações são as mesmas do ruído branco. A primeira compara o *Teste 5.2.2.1 – Caso A*, reconhecedor treinado em modo "clean" e adição de ruído babble ao conjunto de teste, e o *Teste 5.2.2.2 – Caso A*, onde a única diferença em relação ao caso anterior consiste na utilização de Speech Enhancement no conjunto de teste. Os gráficos seguintes ilustram esta comparação.

Cenário 1 – Controlo de cadeira de rodas

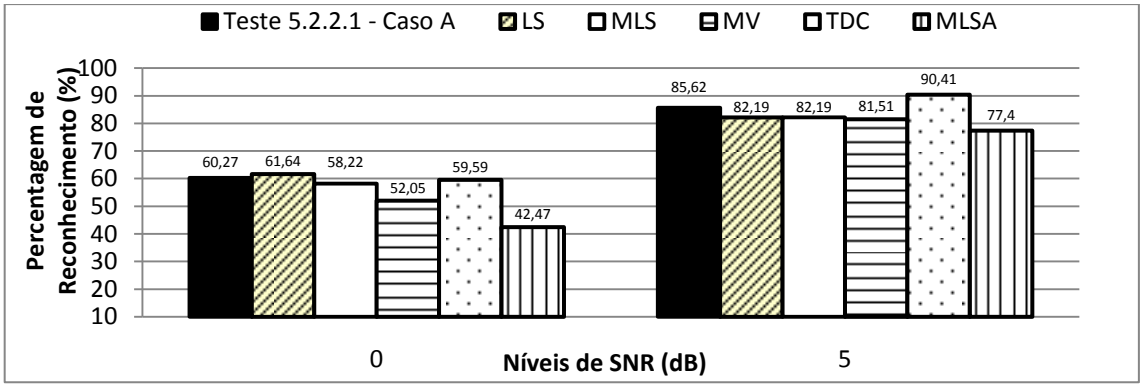

Figura 42 – Efeito de Speech Enhancement no desempenho do reconhecedor (Cenário 1 – Caso A – Ruído Babble)

Cenário 2 – Controlo de sala de cinema

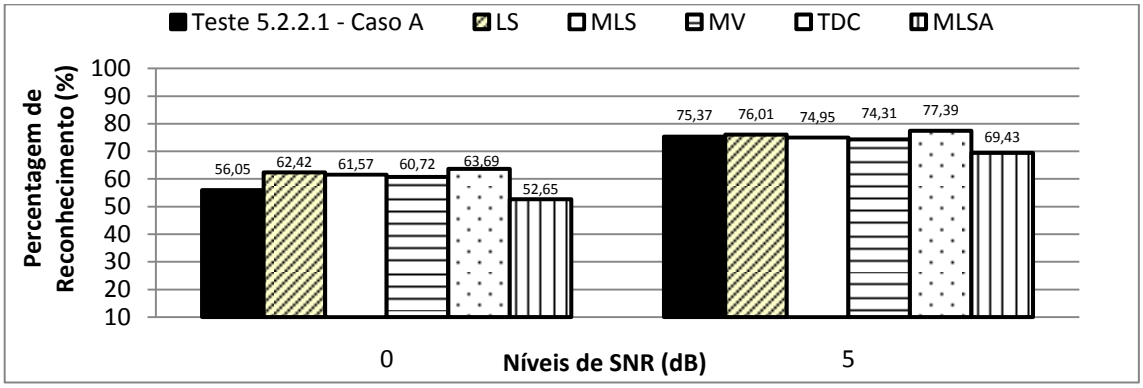

Figura 43 – Efeito de Speech Enhancement no desempenho do reconhecedor (Cenário 2 – Caso A – Ruído Babble)

Analisando os gráficos, no primeiro cenário e para um SNR de 0dB, concluímos que apenas o método ***LS*** melhora o desempenho do reconhecedor. O mesmo acontece para um SNR de 5 dB, mas neste caso o método que se destaca é o ***TDC***.

No segundo cenário, com SNR igual a 0 dB, à excepção do ***MLSA***, todos os outros apresentam percentagens de reconhecimentos superiores, quando comparados com o caso em que não é utilizado Speech Enhancement. Para 5dB, os resultados são globalmente muito parecidos, salientando-se os métodos ***LS*** e ***TDC***.

A mesma comparação foi feita para o caso B. Comparamos o *Teste 5.2.2.1 – Caso B*, onde foi adicionado ruído ao conjunto de treino e teste, e o *Teste 5.2.2.2 – Caso B*, equivalente ao anterior, acrescentando-se Speech Enhancement ao conjunto de teste. Os resultados obtidos são apresentados nas figuras seguintes.

Cenário 1 – Controlo de cadeira de rodas

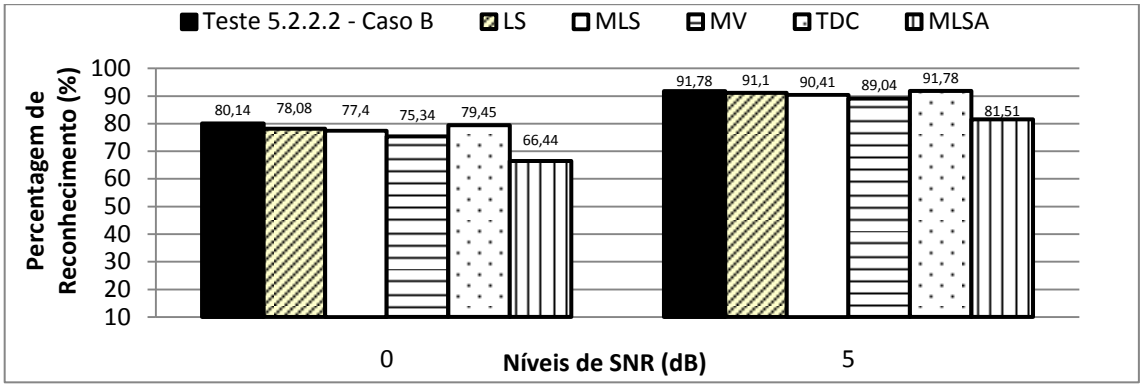

Figura 44 – Efeito de Speech Enhancement no desempenho do reconhecedor (Cenário 1 – Caso B – Ruído Babble)

Cenário 2 – Controlo de sala de cinema em casa

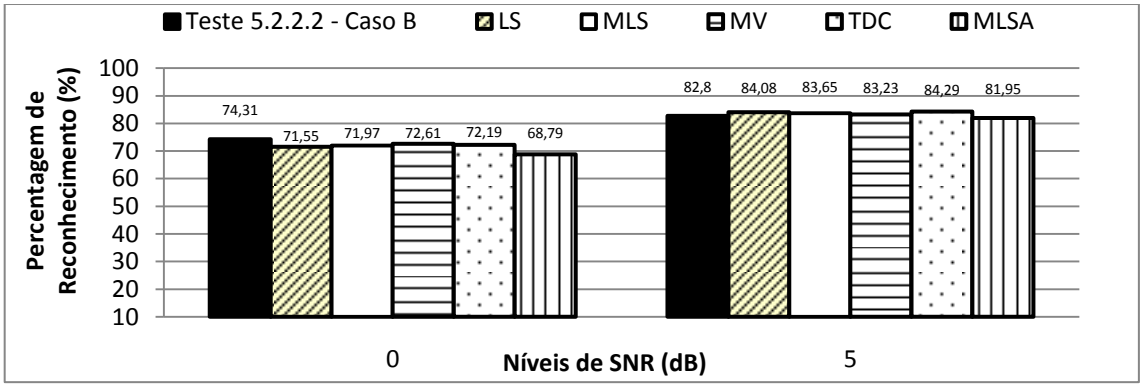

Figura 45 – Efeito de Speech Enhancement no desempenho do reconhecedor (Cenário 2 – Caso B – Ruído Babble)

Pela análise gráfica, para um SNR de 0dB e em ambos os cenários, não existe nenhum método que tenha melhor desempenho relativamente ao caso em que não é utilizado Speech Enhancement no conjunto de teste. Com SNR igual a 5dB, no primeiro cenário destacamos o método ***TDC***, enquanto no segundo cenário, com a excepção do método ***MLSA***, todos os outros métodos melhoram o desempenho do reconhecedor.

5.3. Efeito de variação do limiar

Este tema foi meramente exploratório e tem por objectivo verificar se o aumento do limiar influencia a percentagem de reconhecimento. A título de exemplo, fez-se variar o valor do limiar para 90%, verificando-se os resultados obtidos para um determinado teste dos realizados anteriormente. A escolha foi feita com base nos resultados obtidos para o limiar de 80%.

Assim, escolheu-se o caso em que o reconhecedor base foi treinado adicionando-se os dois tipos de ruído referidos, branco e babble. Ao conjunto de teste foi adicionado também ruído e aplicado Speech Enhancement. Os resultados apenas foram obtidos para o cenário 1, controlo de uma cadeira de rodas.

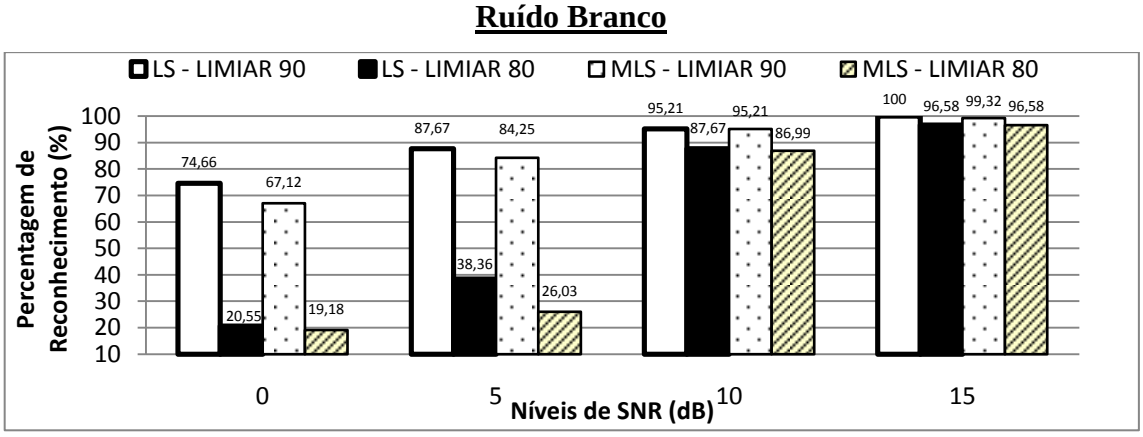

Figura 46 – Efeito de variação do limiar (Método LS e MLS – Ruído Branco)

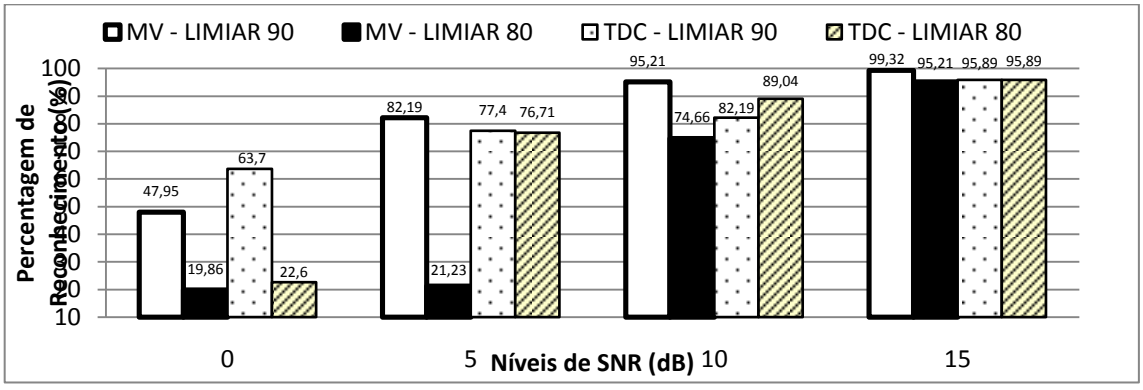

Figura 47 – Efeito de variação do limiar (Método MV e TDC – Ruído Branco)

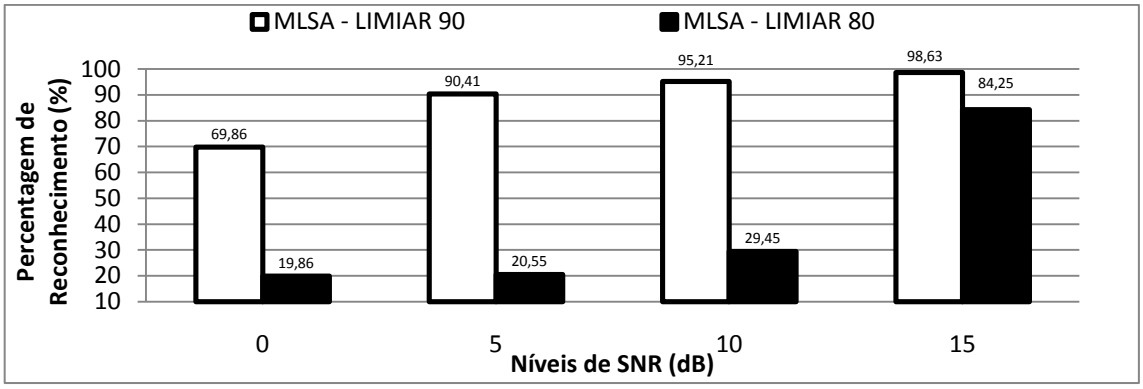

Figura 48 – Efeito de variação do limiar (Método MLSA – Ruído Branco)

A variação do limiar para 90% melhora o desempenho do reconhecedor, sendo os melhores casos para níveis de SNR mais baixos, onde a discrepância de resultados é mais acentuada. Observando os gráficos podemos ainda verificar que o método ***MLS***, muito equiparado ao ***MLSA***, é o que apresenta percentagens de reconhecimento mais elevadas, quando o limiar varia para 90%.

Ruído Babble

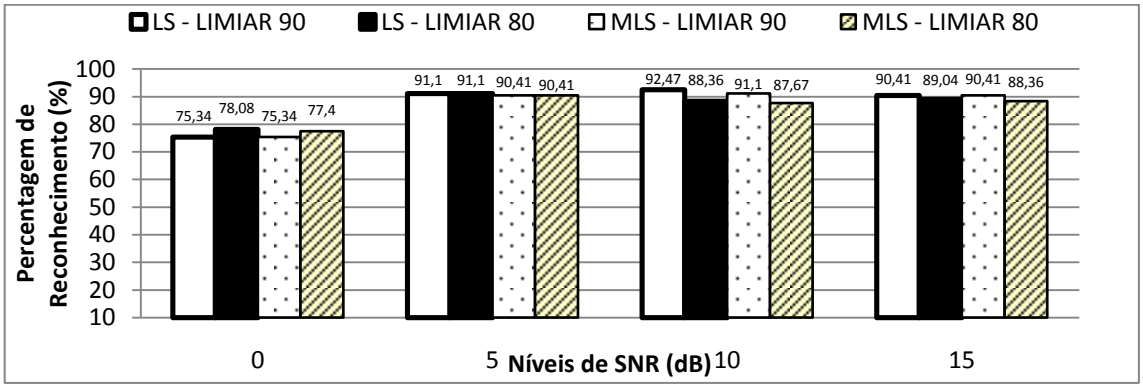

Figura 49 – Efeito de variação do limiar (Método LS e MLS – Ruído Babble)

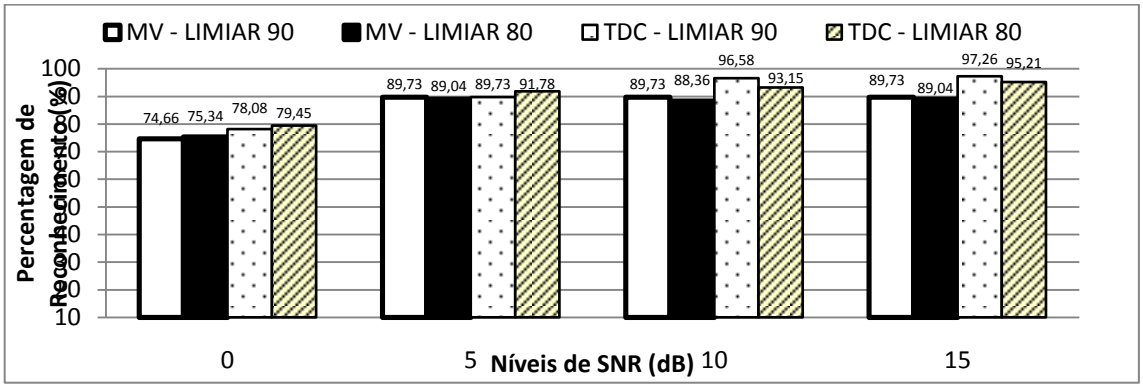

Figura 50 – Efeito de variação do limiar (Método MV e TDC – Ruído Babble)

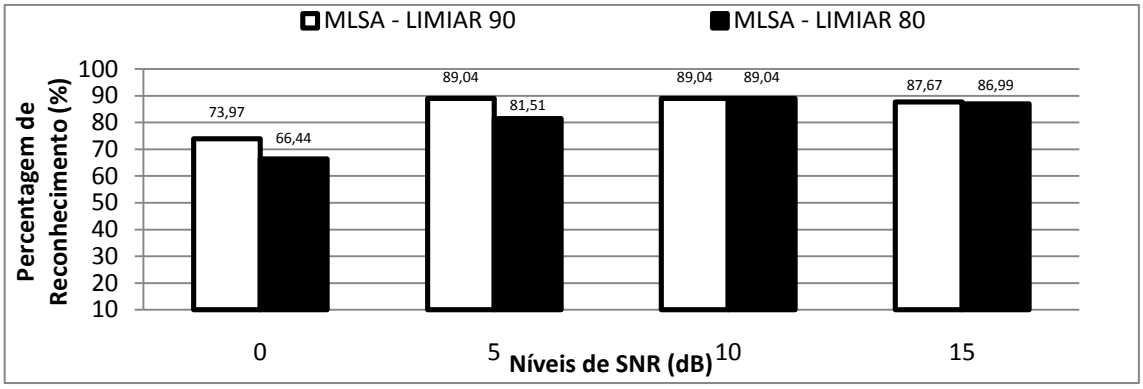

Figura 51 – Efeito de variação do limiar (Método MLSA – Ruído Babble)

Ao contrário do ruído branco, neste caso a discrepância de resultados com a variação do limiar não é tão acentuada, para os vários níveis de SNR. Concluímos ainda que o método mais eficaz é o ***TDC***.

5.4. Efeito de redução de ruído pelo processamento dos coeficientes MFCC e Delta

Passamos agora a outra fase de testes, onde a redução do ruído referida na secção 3.5.2 do capítulo 3 é aplicada aos coeficientes MFCC e Delta primeiramente extraídos dos sinais de áudio utilizados. Podemos destacar dois casos diferentes na aplicação desta metodologia.

No primeiro caso (Caso A), a técnica referida é aplicada apenas aos 13 coeficientes mfcc extraídos, e só de seguida se procede ao cálculo dos coeficientes delta associados a estes. Como segundo caso (Caso B), a reconstrução é aplicada a todos os coeficientes extraídos (mfcc e delta). O limiar definido foi de 80%.

5.4.1. **Comparação entre os dois casos. Conjunto de treino e teste sem adição de ruído**

Nesta fase de testes não foi adicionado qualquer tipo de ruído aos dois conjuntos de sinais áudio, aplicando-se a reconstrução dos coeficientes mfcc e delta aos dois conjuntos, treino e teste, segundo os dois casos enunciados.

Nas tabelas seguintes apresentam-se os resultados para ambos os cenários.

Cenário 1 – Controlo de cadeira de rodas

Método de Speech Enhancement	*Caso A*	*Caso B*
Least Squares	91.10%	91.10%
Modified Least Squares	90.41%	91.10%
Minimum Variance	89.73%	90.41%
Time-Domain Constraint	92.47%	91.10%
MLSA	86.99%	88.36%

Tabela 4 – Efeito de redução de ruído pelo processamento dos coeficientes MFCC e Delta (Cenário 1)

Cenário 2 – Controlo de sala de cinema em casa

Método de Speech Enhancement	*Caso A*	*Caso B*
Least Squares	74.95%	74.95%
Modified Least Squares	74.10%	74.73%
Minimum Variance	73.89%	74.52%
Time-Domain Constraint	76.65%	75.80%
MLSA	66.45%	70.49%

Tabela 5 – Efeito de redução de ruído pelo processamento dos coeficientes MFCC e Delta (Cenário 2)

Para o primeiro cenário, no caso A verificamos que o método ***TDC*** é o mais eficaz, enquanto no caso B existem 3 métodos com desempenho equivalente ***LS, MLS e TDC.*** No segundo cenário, o método ***TDC*** destaca-se em relação aos restantes, para ambos os casos.

5.4.2. **Efeito de adição de ruído aos dois conjuntos de sinais áudio**

Na presente secção pretendemos testar o efeito do factor ruído na adição aos sinais áudio dos conjuntos de treino e teste. Como tal, foram adicionados os dois tipos de ruído, branco e babble, aos dois conjuntos de treino, e aplicado o método. Nesta secção, apenas foram processados resultados para o primeiro cenário.

Nos gráficos seguintes apresentamos a percentagem de reconhecimento do reconhecedor base em função dos níveis de SNR utilizados (0, 5, 10 e 15 dB), comparando-se os dois casos enunciados na secção anterior (Caso A e B), para cada método definido (LS, MLS, MV, TDC e MLSA).

Ruído Branco

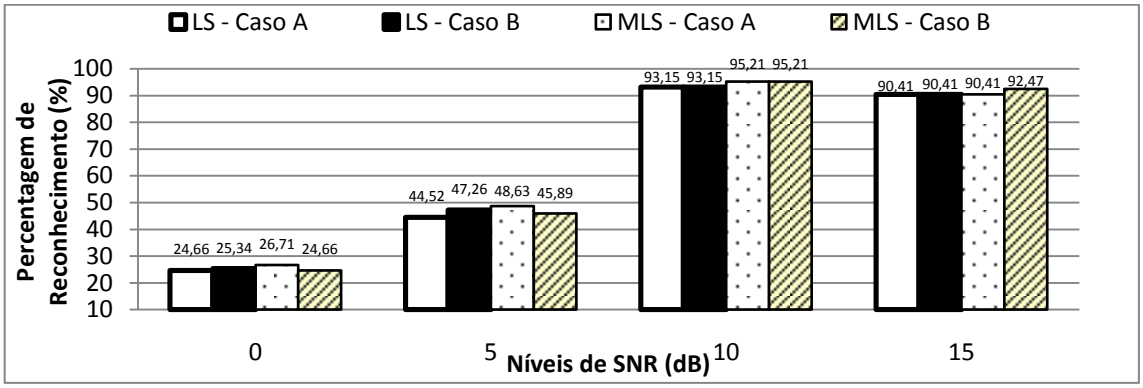

Figura 52 – Efeito de redução de ruído branco pelo processamento dos coeficientes MFCC e Delta (Método LS e MLS)

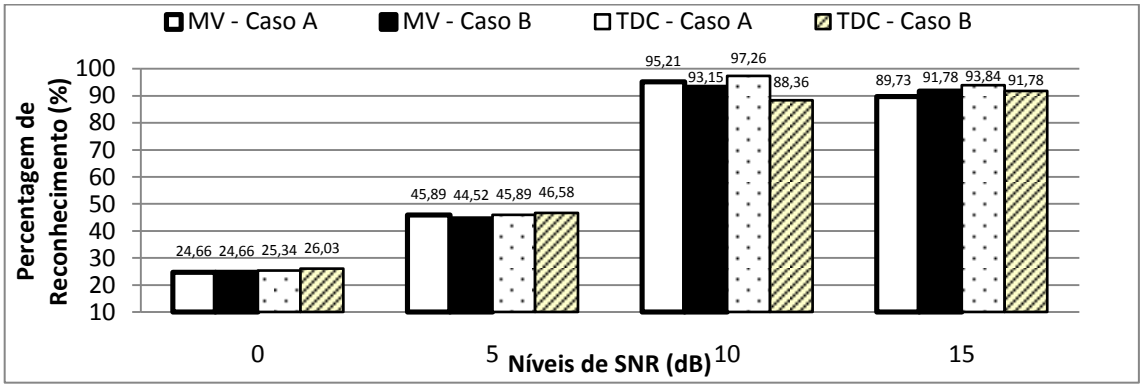

Figura 53 – Efeito da redução de ruído branco pelo processamento dos coeficientes MFCC e Delta (Método MV e TDC)

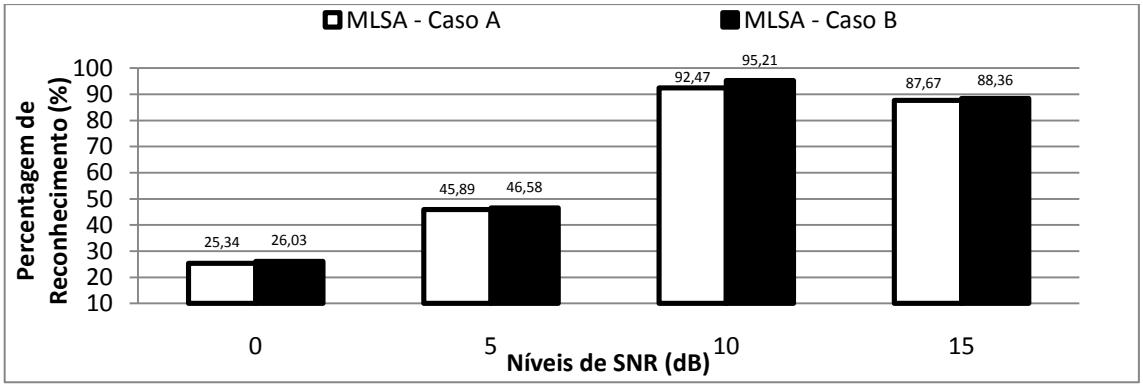

Figura 54 – Efeito da redução de ruído branco pelo processamento dos coeficientes MFCC e Delta (Método MLSA)

Com a adição de ruído, verificamos que os resultados são inferiores aos registados na secção anterior. Observando os gráficos, concluímos que a percentagem de reconhecimento para os vários métodos, em ambos os casos, é semelhante. No caso A, para níveis de SNR baixos, o método ***MLS*** apresenta melhor desempenho, enquanto em níveis superiores é o método ***TDC*** que se destaca. Analisando o caso B, verificamos que o método ***MLS*** tem globalmente, ou seja, para os vários níveis de SNR, um desempenho superior, quando comparado com os restantes métodos, não sendo esta diferença muito acentuada.

Ruído Babble

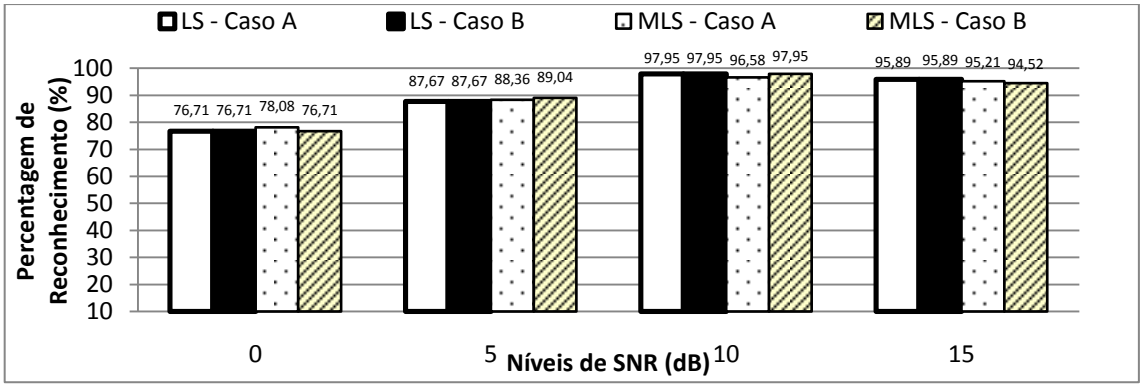

Figura 55 – Efeito de redução de ruído babble pelo processamento dos coeficientes MFCC e Delta (Método LS e MLS)

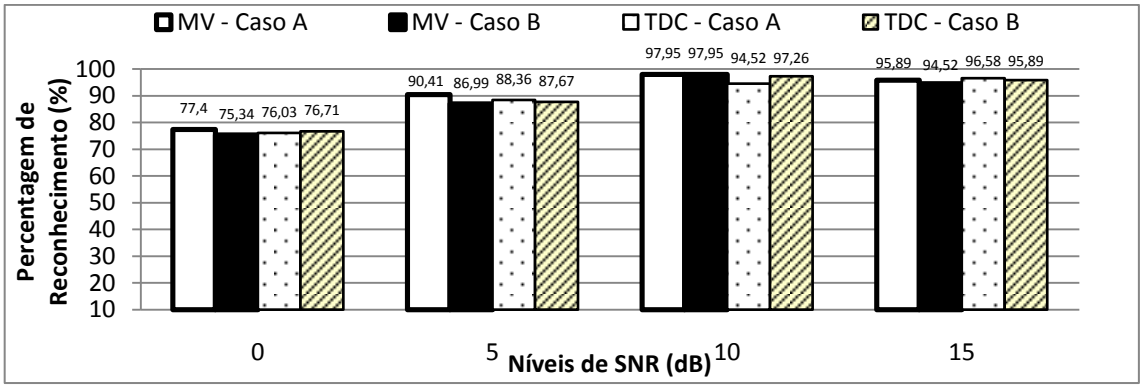

Figura 56 – Efeito de redução de ruído babble pelo processamento dos coeficientes MFCC e Delta (Método MV e TDC)

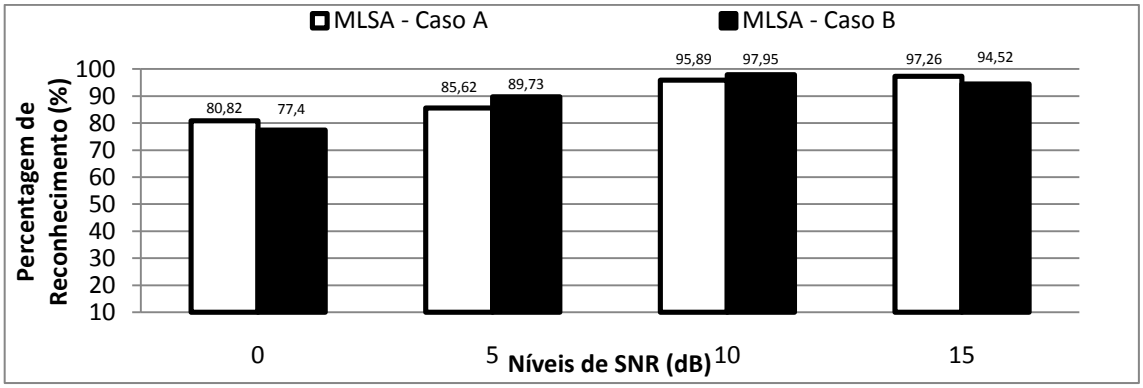

Figura 57 – Efeito de redução de ruído babble pelo processamento dos coeficientes MFCC e Delta (Método MLSA)

Comparando o ruído babble com o ruído branco, verificamos que o primeiro continua a apresentar percentagens de reconhecimento superiores para os vários métodos e níveis de SNR. No caso A, destacamos o método ***MLSA*** como o que globalmente demonstra melhor desempenho. No caso B, o método ***MLSA*** volta a salientar-se, apesar de os resultados serem idênticos entre os vários métodos.

Capítulo 6 **Conclusão**

6.1. Resumo do trabalho realizado

Durante a realização desta dissertação surgiram alguns contratempos, que com maior ou menor dificuldade foram sendo ultrapassados. Foi necessário fazer um estudo sobre o tema da dissertação, recorrendo a vários documentos relacionados com os mais variados assuntos, tais como ***coeficientes mfcc***, ***reconhecedores de fala, métodos de Speech Enhancement baseados em SVD***, entre outros.

O primeiro passo foi aprender a metodologia correcta para a extracção dos coeficientes mfcc de um dado sinal de voz. De seguida, transpor essa metodologia para código Matlab, criando-se uma função cujo resultado eram ficheiros com extensão mfc, que continham os referidos coeficientes. O passo seguinte passou a ser o método de decomposição baseado em SVD, de modo a reconstruir o sinal de voz da melhor forma possível. Tal como no caso anterior, após a aprendizagem teórica, foi necessário transpor para código o referido método. Como passo final temos a criação de um reconhecedor de fala robusto, dependente do orador e baseado em HTK, começando a surgir os principais *"bugs"*.

Um dos principais relaciona-se com o desempenho do reconhecedor base aquando da utilização da primeira versão da função Matlab desenvolvida, que permitia extrair os ***coeficientes mfcc*** dos sinais de áudio dos dois conjuntos, treino e teste. Assim, na primeira versão desta função, o desempenho do reconhecedor era bastante medíocre, quando comparado com o desempenho do comando HCopy do software para criação de reconhecedores de fala, HTK. Por esta razão, surgiu a necessidade de uma reprodução mais fiel do código do comando HCopy [17], recorrendo-se à sua implementação em código C, para que o nosso reconhecedor base produzisse um desempenho mais satisfatório e próximo deste.

Na criação do reconhecedor de fala também surgiram algumas dificuldades, pois o software em utilização era uma novidade, sendo necessário maior empenho. Por outro lado, a criação do reconhecedor de fala exigiu o recurso a uma nova linguagem de

programação até aí desconhecida, linguagem *Perl*. No entanto, a sua aprendizagem tornou-se simples e bastante útil no desenvolvimento do sistema implementado. O resultado desta dissertação é a criação de um reconhecedor de fala que utilize as metodologias referidas, de maneira a verificar o seu desempenho nas mais variadas situações.

6.2. Principais resultados e conclusões

Os primeiros resultados obtidos remontam à comparação entre a função criada para a extracção dos coeficientes mfcc e a ferramenta HCopy do HTK. Concluímos que a função criada tem um bom desempenho, pois os resultados obtidos são da mesma ordem de grandeza, quando comparados com o desempenho da ferramenta HCopy. Por outro lado, com os testes realizados sobre o factor ruído no desempenho do reconhecedor, verificamos que a ***adição de ruído, branco ou babble, ao conjunto de treino é benéfica para a melhoria da percentagem de reconhecimento***. A única excepção ocorre para o ruído branco, quando ***é utilizado Speech Enhancement no conjunto de teste***, verificando-se que ***não é imprescindível treinar o reconhecedor adicionando ruído***.

Na comparação de resultados obtidos entre a adição de ruído branco e babble, observamos que as percentagens de reconhecimentos do segundo caso são, globalmente, superiores. A razão deste facto está associada à forma como a adição de ruído é feita em cada um dos casos. Para o ruído babble, o ruído a adicionar aos sinais áudio é sempre o mesmo, enquanto no ruído branco, como recorremos ao comando *awgn* do Matlab, os valores de ruído criados não são sempre os mesmos, existindo assim um erro associado.

Como ponto fulcral deste trabalho, temos os resultados obtidos na utilização de Speech Enhancement no conjunto de sinais de teste. A comparação foi feita utilizando ruído branco e ruído babble, observando-se que nem sempre a utilização de Speech Enhancement é útil para a melhoria do desempenho do reconhecedor. No caso em que o reconhecedor é treinado em ***modo "clean"*** e ***adicionando ruído branco*** ao conjunto de teste, a utilização de Speech Enhancement apenas ***é benéfico para um SNR de 5dB***. Se for ***adicionado ruído branco aos dois conjuntos***, treino e teste, a utilização de Speech Enhancement ***não melhora*** o desempenho do reconhecedor. Na análise para ruído babble,

quando o reconhecedor é treinado em ***modo "clean"*** e ***adicionando ruído babble***, existe pelo menos um método de Speech Enhancement, para ambos os cenários e ambos os níveis de SNR, que ***melhora*** a funcionalidade do reconhecedor. Quando é ***adicionado ruído aos dois conjuntos***, apenas para ***SNR igual a 5 dB***, a utilização de Speech Enhancement ***é benéfica***. Para um SNR de 0dB, apesar de não existir um método de Speech Enhancement que se evidencie, a diferença de percentagem de reconhecimento não é muito significativa.

Quantos aos dois casos exploratórios, ***variação de limiar*** e ***redução de ruído pelo processamento dos coeficientes mfcc e delta***, os resultados são promissores, podendo ser alvo de estudo futuro.

Concluímos que ***a introdução do método de Speech Enhancement implementado no conjunto de teste***, apesar de não ser eficaz em todas as condições, é possível ser utilizado de forma a melhorar o desempenho do reconhecedor, mesmo estando este previamente treinado.

6.3. Sugestões de continuação

Como principais sugestões de continuação deverão ser salientados os dois casos de estudo exploratórios apresentados nesta dissertação: o *efeito de variação do limiar* e o *efeito de redução de ruído pelo processamento dos coeficientes MFCC e Delta*. No primeiro caso, foi possível observar que o aumento do limiar pode ser benéfico para o desempenho de um reconhecedor, sugerindo-se um estudo futuro que varie "passo a passo" o limiar entre os 80% e os 90%, por exemplo, verificando-se cuidadosamente o efeito deste factor no seu desempenho. Os resultados obtidos para o segundo caso de estudo referido, não descartam uma possibilidade de continuação do estudo, podendo ser uma vertente útil para melhorar o desempenho de um reconhecedor.

Podemos ainda referir a automatização do reconhecedor de fala e, a implementação de outras técnicas de Speech Enhancement.

Referências

[1] Young, Steve, et al. *The HTK Book*. Cambridge University Engineering Department, 2001.

[2] You, Chang Huai, Rahardja, Susanto e Koh, Soo Ngee. *Audible Noise Reduction in Eigendomain for Speech Enhancement*. IEEE Transactions on Audio, Speech and Language Processing, 2007, Vol. XV.

[3] www.ieeta.pt/~ajst/pdv, Novembro 2008.

[4]Shlens, Jonathon. *A Tutorial on Principal Component Analysis.* University of California, 2005, Vol. II.

[5] www.auditory.org/mhonarc/2006/msg00609.html, Fevereiro 2009.

[6] Ribeiro, Carlos Eduardo de Meneses. *Processamento Digital de Fala*. 2002.

[7] Kacur, Juraj e Rozinaj, Gregor. *Pratical Issues of Building Robust HMM Models Using HTK and SPHINX Systems.* Slovak University of Technology, Faculty of Electrical Engineering and Information Technology.

[8] Hu, Yi e Loizou, Philipos C. *Evaluation of Objective Quality Measures for Speech Enhancement* . IEEE Transactions on audio, speech, and language processing, 2008, Vol. XVI.

[9] Hansen, Per Christian e Jensen, Soren Holdt. *Subspace-Based Noise Reduction for Speech Signals via Diagonal and Triangular Matrix Decompositions: Survey and Analysis.* EURASIP Journal on Advances in Signal Processing, pp. 20-40, 2007.

[10] Gail Ether, Ph.D. *Speech recognition with built in noise and chatter immunity.* IC Tech, Inc, 2005.

[11] Benesty, Jacob, Makino, Shoji e Chen, Jingdong. *Speech Enhancement.* 2005.

[12] Abreu, Carlos Jorge Enes Capitão de. *Interface com Reconhecimento de Fala para Apoio a Pessoas com Limitações Funcionais.* Universidade de Aveiro, 2007.

[13] labrosa.ee.columbia.edu/matlab/rastamat/mfccs.html.

[14] Manual Matlab.

[15] *Grande Biblioteca Multilingue - Dicionário Português/Alemão e Alemão/Português.* Porto Editora, 2002.

[16] www.google.com/codesearch

[17] HTK Speech Recognition Software, 2001.

[18] Silva, Tomás Oliveira e. *Apontamentos de Processamento Digital de Sinal.* 2009-2010.

[19] Jensen, Soren Holdt, et al. *Reduction of Broad-Band Noise in Speech by Truncated QSVD.* IEEE transactions on speech and audio processing, 1995, Vol. III.