FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Reorganização espectral de sinais de fala na banda de [0, 2]kHz

Ricardo A. S. Da Costa

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Telecomunicações

Orientador: Prof. Dr. Aníbal J. S. Ferreira

Maio de 2009

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES 2008/2009

A Dissertação intitulada

"REORGANIZAÇÃO ESPECTRAL DE SINAIS DE FALA NA BANDA (0,2) KHZ"

foi aprovada em provas realizadas em 21/Julho/2009

o júri

Presidente Professora Doutora Ana Maria Rodrigues de Sousa Faria de Mendonça
Professora Associada do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Ana Maria Perfeito Tomé
Professora Associada do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Professor Doutor Aníbal João de Sousa Ferreira
Professor Auxiliar do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

Autor - RICARDO ALEXANDRE SILVA DA COSTA

Faculdade de Engenharia da Universidade do Porto

Resumo

Há indivíduos com deficiência auditiva que apresentam uma perda profunda de sensibilidade a partir dos 2 kHz. Este problema tem grandes implicações na reduzida inteligibilidade de sons de fala, com particular destaque para as consoantes. Tirando contudo partido do facto daqueles indivíduos manterem intacta a região de audição das baixas frequências, pretende-se restituir a inteligibilidade da fala através da inserção controlada, na região espectral entre 1.5 kHz a 2 kHz, de informação espectral da região entre 2 kHz a 5 kHz.

Fez-se a análise do estado-da-arte e implementaram-se quatro algoritmos alternativos para que os indivíduos com deficiência profunda a partir de 2 kHz possam, em particular, reconhecer melhor as consoantes no discurso de um orador. Os algoritmos foram alvo de testes de simulação com palavras do Português Europeu. Os resultados obtidos revelaram que nalguns casos há melhoria sensível de inteligibilidade, e indicaram que alguns dos algoritmos implementados merecem ser alvo de mais investigação já que exibem maior potencialidade.

Abstract

There are subjects with high hearing loss that have a profound loss of sensibility above 2kHz. This problem has significant implications on the reduced intelligibility of speech sounds, in particular consonants. However, taking advantage of the fact that the low frequency region is reasonably intact, we aim at restoring the intelligibility of speech through the controlled insertion in the spectral region between 1.5 kHz and 2kHz of information pertaining to the spectral region between 2kHz and 5kHz.

The state-of-the-art was reviewed and four alternative algorithms were implemented that attempt to improve the intelligibility of certain speech sounds, namely consonants, when profound hearing loss above 2 kHz is assumed. The algorithms were simulated and tested with European Portuguese words. The obtained results show that in some cases there is significant improvement in intelligibility, and indicate that some of the implemented algorithms should be subject to further research as they exhibit an interesting potential.

Agradecimentos

Em primeiro lugar queria agradecer e dedicar este trabalho a minha esposa Carla, porque sem o seu apoio ao longo destes cinco anos eu não estaria aqui a escrever, e por ter acreditado em mim quando me incentivou a vir cumprir um sonho (não direi de criança mas de adolescente). Não poderia deixar de agradecer ao resto da minha família que também acreditou em mim e me apoiou, aos meus pais José e Teresa, ao meu sogro Leandro, ao meu irmão Santiago e sua esposa Carina, e a minha cunhada Célia.

Gostaria de agradecer ao Prof. Dr. Aníbal Ferreira pela sua atenção sempre que foi por mim solicitado e ao que aprendi com a sua pessoa. Ao colega de laboratório (I322) Ricardo pela sua atenção e disponibilidade.

A todos obrigado e muitas felicidades para o futuro.

O Autor

"If builders built buildings the way programmers wrote programs,
then the first woodpecker that came along would destroy civilization."

"Tell a man there are 300 billion stars in the universe and he'll believe you.
Tell him a bench has wet paint on it and he'll have to touch to be sure.
Great discoveries are made by mistake. "

Murphy´ s Laws

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

FFT	Fast Fourier Transform
C/a/	Consoante e Vogal *a*
/b/V/t/	Consoante *b*, Vogal e Consoante *t*
CV	Consoante Vogal
VC	Vogal Consoante
VCV	Vogal Consoante Vogal
CVC	Consoante Vogal Consoante
CNC	Consoante Vogal Nuclear e Consoante
DR	Dead Region - região morta da cóclea
AFI	Alfabeto Fonético Internacional
F0	Frequência Fundamental

Capítulo 1

Introdução

1.1 Enquadramento

A perda auditiva neurosensorial, também conhecida por surdez nervosa (*nerve deafness*), resulta de problemas na parte interna do ouvido, na qual o nervo que realiza a interface para o cérebro está danificado. Pode ser devido a uma deficiência genética ou desencadeada por diversos factores, tais como a velhice, exposição prolongada em ambientes ruidosos, doses elevadas de medicamentos, consumo de drogas, doenças, ou acidentes.

Apresenta diversos sintomas que a caracterizam como por exemplo a perda da sensibilidade na interpretação da intensidade, a distorção e a menor inteligibilidade dos sons. Há também outros sintomas como por exemplo ,os tons agudos tornam-se menos audíveis, sons como /s/, /f/, e /z/ deixam de ser perceptíveis, a dificuldade de percepção do discurso quando existe ruído de fundo, o aparecimento de uma espécie de zumbido (tinitus) e vertigens.

Os efeitos práticos deste tipo de perda auditiva é a alteração da percepção do som de uma maneira complexa e não-linear. As amplitudes de harmónicas do sinal de voz e modulações temporais são algumas das características do som que são alteradas, resultando numa distorção ao nível do timbre e da sonoridade. Estas alterações afectam a inteligibilidade e a qualidade do som. As características que mais prejudicam a inteligibilidade do som são:

i Redução da sensibilidade;

ii Deslocamento não-linear da sonoridade;

iii Reduzida selectividade de frequências;

iv Resolução temporal reduzida.

Com base nestes conhecimentos, o processamento digital de sinal pode ter um papel na compensação dos inconvenientes provocados por este tipo de perda de audição, e melhorar substancialmente a inteligibilidade na percepção do som, com particular interesse o som da fala.

1.2 Caracterização do Problema

O problema que indivíduos com perda auditiva enfrentam no seu dia a dia, como foi referido em 1.1, é a perda de reconhecimento de certas palavras durante o discurso de um orador. Esta perda resulta do facto do nosso aparelho fonador produzir diversos tipos de sons, que são estruturados de maneira diferente no domínio das frequências, tal como será referido no capítulo 3.

Este trabalho concentra-se nos casos em que a perda auditiva é profunda acima dos 2 kHz. Neste casos, as próteses auditivas actuais não conseguem dar resposta ao problema visto que estas são vocacionadas para indivíduos com perdas moderadas. Em concreto, uma das técnicas que estes aparelhos empregam é a amplificação do sinal na zona da perda auditiva, mas aplicar um ganho muito elevado às altas frequências pode resultar num grande desconforto para além de não resolver o problema fundamental.

1.3 Objectivos

Pretende-se neste trabalho fazer prova do conceito de uma solução de análise e síntese que permita concentrar a espectro relevante de sinais de fala, que tipicamente ocupa uma largura de banda de cerca de 5 kHz, numa banda estreita de apenas 2 kHz, de modo a melhorar a inteligibilidade daqueles sinais por pacientes com perdas auditivas profundas a partir de 2 kHz.

O trabalho visa a investigação, implementação e validação de uma solução de processamento de sinal que permita melhorar significativamente a inteligibilidade de sons naturais, com destaque para sinais de fala, por pessoas com perdas auditivas profundas a partir dos 2 kHz. Em termos gerais a solução consiste em reorganizar o espectro típico de voz natural cuja largura de banda é cerca de 5 kHz, na banda estreita entre 0 e 2 kHz.

1.4 Estrutura da Dissertação

Para além deste primeiro capítulo que serve de introdução do trabalho, a Dissertação é organizada em mais quatro capítulos, as quais passamos a descrever de seguida.

Capítulo 2 introdução ao sistema auditivo humano, proporcionado uma visão geral da sua estrutura interna e do seu funcionamento.

Capítulo 3 exposição dos tipos de sons do Português Europeu, como são produzidos e as suas propriedades.

Capítulo 4 contém a descrição do estado da arte, evidenciando o que se tem feito nesta área com destaque para as principais técnicas envolvidas e resultados alcançados.

Capítulo 5 apresenta o algoritmo e as simulações realizadas *offline* em ambiente Matlab.

Capítulo 6 apresentação de conclusões e o trabalho futuro.

Capítulo 2

Sistema Auditivo Humano

2.1 Introdução

Neste capítulo serão abordados conceitos básicos sobre o sistema auditivo humano, permitindo assim explicitar de um ponto de vista geral o funcionamento do sistema, com destaque para a selectividade espectral, sem contudo aprofundar visto que não é essa a essência do trabalho.

2.2 Estrutura Interna do Sistema Auditivo Humano

O sistema auditivo humano é composto basicamente por três partes, sendo elas o ouvido externo, o ouvido médio e o ouvido interno Ferreira et al. (2009). A figura 2.1 apresenta a estrutura do sistema. O ouvido externo consiste na parte responsável pela interface entre a pressão do ar

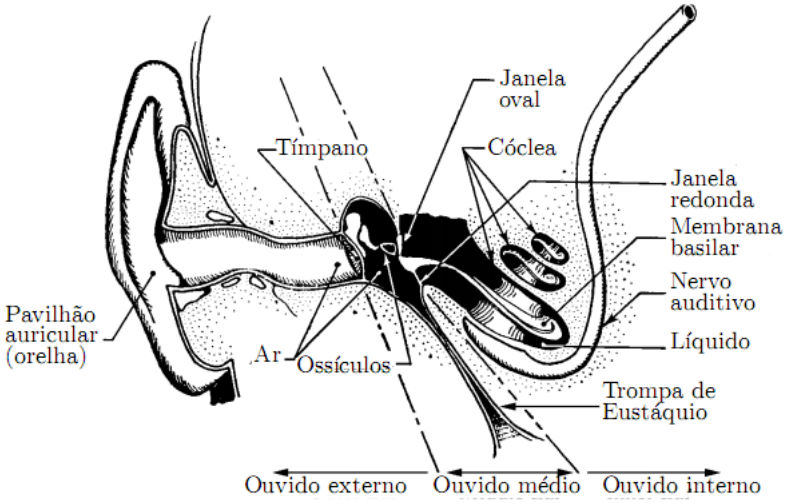

Figura 2.1: Estrutura do sistema auditivo humano (adaptada de Ferreira et al. (2009)).

criada pelo deslocamento das ondas (abordado em 3.5) e a decorrente energia mecânica transmitida através do tímpano e ossículos Ferreira et al. (2009). De facto o ouvido externo é constituído a montante pela orelha que capta os sons. Estes são encaminhados através do canal auditivo para o tímpano, que por sua vez realiza a tarefa de converter a pressão acústica em energia mecânica colocando os tímpanos e ossículos em vibração. O ouvido médio tem como tarefa comunicar a energia mecânica oferecida pela primeira secção do sistema ao ouvido interno, através de três ossículos ligados mecanicamente entre si, que são martelo, bigorna e estribo. Estes ossículos têm por função realizar uma adaptação de impedâncias entre os dois meios. A janela oval situada no início do ouvido interno, capta a energia proveniente do ouvido médio, e transmite esta energia através de um liquido que preenche os três canais da cóclea. Os três canais são paralelos, separados por duas membranas, incluindo a membrana basilar e enrolados em forma de caracol. Estes canais são designados por rampa vestibular, ducto coclear e rampa timpânica. A membrana basilar desempenha um papel fundamental, já que aloja células ciliadas ao longo da sua extensão (cerca de 35 mm Ferreira et al. (2009)), cujas propriedades físicas variam significativamente, o que lhe permite possuir características de ressonância para as diversas frequências de excitação sonora. Dessa forma podemos equiparar esta membrana a um analisador espectral. A figura 2.2 exempli-

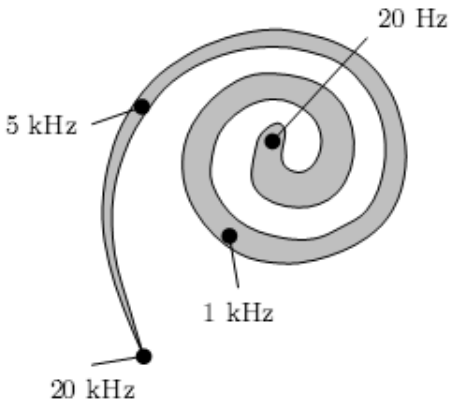

Figura 2.2: Exemplo de uma membrana basilar com as diferentes localizações de frequências de ressonâncias (adaptada de Ferreira et al. (2009)).

fica diversas localizações de frequências de ressonâncias. A ressonância provoca uma cadência de impulsos nervosos (sintonia tonotópica) que são transportados pelos nervos auditivos para o cérebro.

Para além da tarefa principal de adaptação de impedâncias os ossículos também têm uma função de protecção, ou seja, quando o nível de intensidade acústica for perigoso, eles atenuam o sinal através dos músculos que os interligam. O ouvido médio está ligado ao exterior através da trompa Eustáquio para assim poder manter um equilíbrio de pressões nos dois lados do tímpano. No ouvido interno a janela redonda compensa o esforço criado pelos ossículos (mais concretamente pelo estribo) na janela oval com um movimento de sentido contrário.

2.3 Audição

O som apresenta características próprias (tópico abordado em 3.5) que permitem que ele seja ouvido, a frequência e a intensidade são as principais. Tipicamente o nosso ouvido reage as variações da intensidade acústica de maneira não-linear, ou seja, pode ser representada por uma escala logarítmica Ferreira et al. (2009). A unidade utilizada para quantificar o som é o dB SPL (deciBel Sound Pressure Level) e é obtida através da relação logarítmica entre a pressão do sinal em causa e a pressão de referência 20 µPa. A figura 2.3 apresenta um esboço da nossa gama de

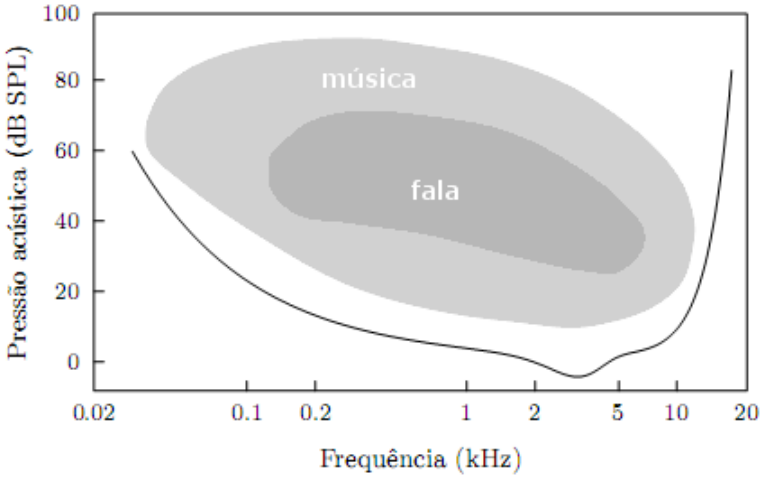

Figura 2.3: Gamas de frequências que o nosso ouvido consegue identificar (adaptada de Ferreira et al. (2009)).

audição para os sons da fala e da música, assim como, o limiar de audição do ouvido humano que se encontra representado pela curva a preto, e é aproximado por :

$$L = 3.64\left(\frac{f}{1000}\right)^{-0.8} - e^{\left(-0.6\left(\frac{f}{1000}-3.3\right)^2\right)} + 0.001\left(\frac{f}{1000}\right)^4.$$

Da análise da figura podemos ainda constatar que o ouvido apresenta maior sensibilidade na região entre 2 kHz e 5 kHz, pelo contrario abaixo dos 20 Hz (denominados infra-sons) e acima dos 20 kHz (denominados ultra-sons), o ouvida deixa de ter percepção.

O limite acima do qual aparece desconforto e/ou dor é de 100 dB a 120 dB, e pode mesmo gerar perdas auditivas graves e irreversíveis. A exposição durante oito horas não pode ser superior a 85 dB Ferreira et al. (2009), e para cada 5 dB acima de 85 dB o período de exposição deve passar para metade.

O que foi referido até agora é considerada audição monoauricular e consiste em aplicar um som a um único ouvido ou aos dois nas mesmas condições. Em contra partida, se aplicarmos o mesmo som mas em condições diferentes, ou seja, não aplicando o som directamente aos dois ouvidos mas sim de um ponto especifico que cria uma situação de desigualdade no trajecto do som

para os dois ouvidos (ver figura 2.4), passa-se a ter a designada audição biauricular. As diferenças a que os ouvidos ficam submetidos são essencialmente o atraso com que o som é percepcionado

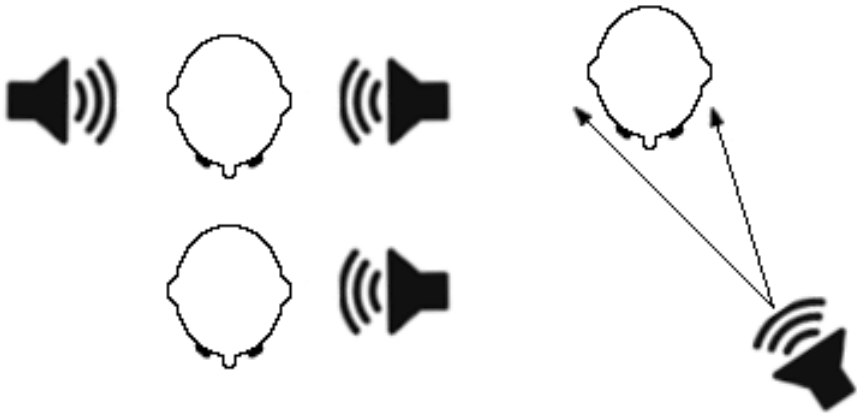

Figura 2.4: O lado esquerdo da figura apresenta as duas configurações do modo monoauricular, por sua vez o lado direito representa o modo biauricular.

por ambos, a intensidade com a qual atinge cada um e o perfil espectral recebido em cada um. Recorrendo as características referidas, o sistema auditivo é capaz de extrair informação sobre a fonte sonora, tais como a localização e a distância Ferreira et al. (2009).

2.4 Bandas Críticas

Devido à constituição interna do sistema auditivo, mais concretamente a cóclea, a percepção em frequência dos sons pode ser comparada à obtida com um banco de filtros sobrepostos Spanias et al. (2007), no qual a resposta em frequência é assimétrica e não linear, e a largura de banda dos filtros não é uniforme pois ela incrementa com o aumento da frequência. Devido a estas características surgiu a necessidade de quantificar esta nova organização de frequências que o sistema auditivo apresenta. A nova escala de frequências foi designada por bandas criticas, tem como unidade o Bark e pode ser calculada Spanias et al. (2007) pela seguinte expressão

$$z(f) = 13\arctan(0.00076f) + 3.5\arctan\left[\left(\frac{f}{7500}\right)^2\right].$$

Por sua vez, a largura de banda que corresponde a um Bark pode ser obtida através de Spanias et al. (2007)

$$BW_c(f) = 25 + 75\left[1 + 1.4\left(\frac{f}{1000}\right)^2\right],$$

e a sua unidade continua a ser o Hertz. A escala de bandas críticas também pode ser apresentada pela tabela 2.1. Da tabela podemos constatar que a banda espectral a que o nosso sistema auditivo é sensível, está dividida em 25 bandas críticas, ou seja, 25 Bark. Em termos anatómicos um Bark

# banda	Frequência central (Hz)	Largura de banda (Hz)
1	50	- 100
2	150	100 - 200
3	250	200 - 300
4	350	300 - 400
5	450	400 - 510
6	570	510 - 630
7	700	630 - 770
8	840	770 - 920
9	1000	920 - 1080
10	1175	1080 - 1270
11	1370	1270 - 1480
12	1600	1480 - 1720
13	1850	1720 - 2000
14	2150	2000 - 2320
15	2500	2320 - 2700
16	2900	2700 - 3150
17	3400	3150 - 3700
18	4000	3700 - 4400
19	4800	4400 - 5300
20	5800	5300 - 6400
21	7000	6400 - 7700
22	8500	7700 - 9500
23	10500	9500 - 12000
24	13500	12000 - 15500
25	19500	15500 -

Tabela 2.1: Escala das bandas críticas (adaptada de Spanias et al. (2007)).

é equivalente a uma distância fixa da membrana basilar, o que por sua vez também corresponde a ter um número constante de células ciliadas Ferreira et al. (2009).

Obtendo a resposta do banco de filtros através de *notch noise*[1], obteve-se a escala de ERB que é uma alternativa à escala de Bark. A largura de banda dos filtros passou a ser caracterizada pela largura de banda equivalente (Equivalent Rectangular Bandwidth (ERB)). Esta escala apresenta uma grande diferença em relação a escala de Bark, já que não pode ser utilizada para valores abaixo e acima da gama dos 100 Hz aos 10 kHz porque só foram medidas as larguras de banda equivalentes para esta gama. A escala ERB pode ser calculada(Härmä et al. (2000)) através de

$$x(f) = 21.3 \log\left(1 + \frac{f}{229}\right),$$

[1] notch noise - consiste numa sinusóide com uma frequência fixa e rodeada de ruído separado do centro da frequência da sinusóide de $2\Delta f$ Rossing (2007)

e as larguras de banda podem ser obtidas(Härmä et al. (2000)) através de

$$BW_{ERB} = 24.7 + 0.108 f_c.$$

A escala de ERB acima dos 500 Hz apresenta uma forma próxima da escala Bark. Além destas duas escalas ainda existe a escala de *mel* que tem como unidade o Mel Ferreira et al. (2009).

Capítulo 3

Fonética do Português Europeu

3.1 Introdução

Este capítulo tem por objectivo oferecer uma visão geral sobre a fonética do Português Europeu, que nos permitirá entender como são produzidos os sons, e quais são as propriedades mais relevantes que nos permitem reconhecê-los.

3.2 Representação

O sistema ortográfico não é um bom método para representar os sons produzidos, isto porque não existe uma relação de um para um quando se escreve e se pronuncia. Devido a isso foi adoptado o Alfabeto Fonético Internacional (AFI) que permite relacionar um símbolo a um som para evitar ambiguidades. Este também tem a vantagem, como decorre do seu nome, de ser internacional, ou seja, um som qualquer será sempre representado pelo mesmo símbolo em qualquer outro lugar do mundo, sempre e quando este som existir numa língua. A conversão para o AFI requer uma regra prática, não basta saber o símbolo que relaciona um som, tem que se delimitar os símbolos por parêntesis rectos []. A tabela 3.1 apresenta os tipos de delimitação. As tabelas

Delimitação	Representação
< . . . >	Ortográfica
/ . . . /	Fonológica
[. . .]	Fonética

Tabela 3.1: Tipos de delimitadores de palavras que nos permitem identificar a sua representação (adaptada de Mateus et al. (2005))

seguintes mostram a relação entre o AFI e os símbolos ortográficos existentes no Português Europeu, a tabela 3.2 mostra qual é símbolo que o som de cada letra pode ter, isto é, uma letra pode ser pronunciada de diferente maneira conforme a exigência da palavra. Por sua vez a tabela 3.3

indica quais os símbolos que são necessários para representar uma junção entre uma vogal e as consoantes n e m. A utilização dos acentos também têm uma correspondência de símbolos no AFI, a tabela 3.4 apresenta-os. Para finalizar os dígrafos que são uma junção de duas letras (pode ser a mesma letra) para criar um som, são apresentados na tabela 3.5. Nas tabelas citadas a coluna da esquerda são os símbolos do sistema ortográfico, a coluna do meio representa os símbolos do AFI, e por fim na última coluna podemos observar exemplos que relacionam as duas colunas anteriores.

3.3 Aparelho Fonador

O sistema responsável pela geração de sons é designado por aparelho fonador, o qual é constituído por pulmões, laringe, e tracto vocal. A figura 3.1 apresenta uma visão detalhada das duas partes mais importantes visto que são responsáveis pelos diferentes tipos de sons, embora os pulmões sejam importantes a sua função é sempre a mesma, que é fornecer o ar necessário para excitar os outros dois.

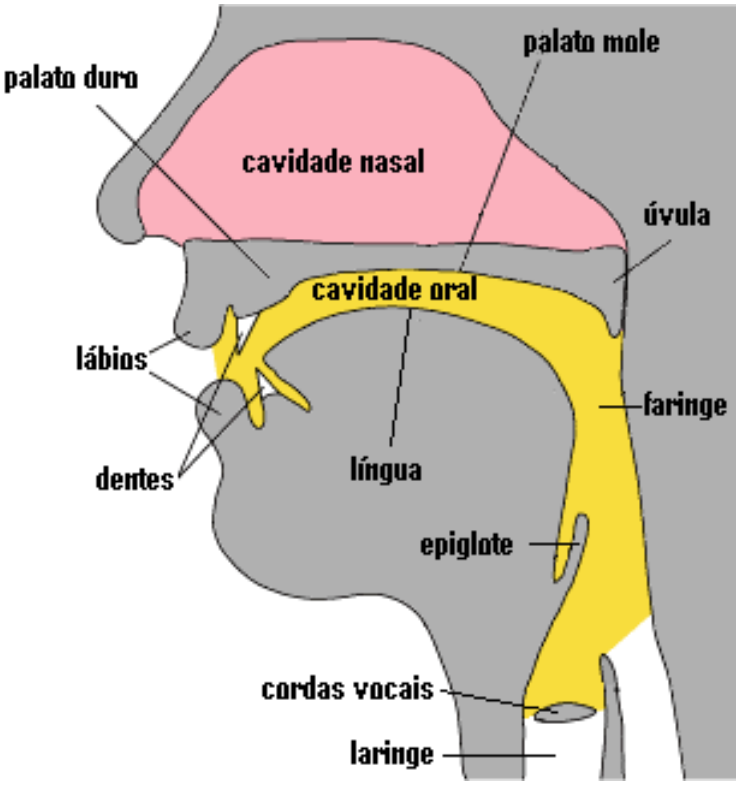

Figura 3.1: Ilustração dos membros que formam o aparelho fonador e que são responsáveis pela criação dos diversos tipos de sons da fala(adaptada de http://www.indiana.edu/~hlw/PhonUnits/vowels.html, último acesso em 06/05/09 e legendagem de Mateus et al. (2005)).

A laringe é uma estrutura composta por músculos, cartilagens, ligamentos, e membranas mucosas, criando assim uma estrutura circular que é atravessada pelas cordas vocais ou também conhecidas por pregas vocais, que por sua vez estão ligadas à cartilagem tiróide. A distância entre

Tabela 3.2: Relação dos símbolos do alfabeto fonético internacional com os símbolos ortográficos (letras) (adaptada de Mateus et al. (2005))

Letras	Sons	Exemplos
<a,A>	[a]	**A**rte, sap**a**to
	[ɐ]	**A**metista, f**a**relo
<b,B>	[b]	**B**atata, ca**b**ide
<c,C>	[k]	**C**aruma, fa**c**a
	[s]	**C**igarra, héli**c**e
<d,D>	[d]	**D**ama, ama**d**or
<e,E>	[ɛ]	Can**e**la
	[e]	Can**e**ta
	[ɨ]	P**e**dal
	[j]	M**e**ada
	[ɐ]	Esp**e**lho
	[j̃]	Mã**e**
	[i]	**E**
<f,F>	[f]	**F**amília, a**f**inador
<g,G>	[g]	**G**avião, apa**g**ador
	[ʒ]	**G**irafa, má**g**ico
<i,I>	[i]	**I**greja, s**i**nal
	[j]	**I**ate, ga**i**vota
<j,J>	[ʒ]	**J**anela, poe**j**os
<l,L>	[l]	**L**aranja, ma**l**a
	[ɫ]	A**l**tura
<m,M>	[m]	**M**orango, ca**m**élia
<n,N>	[n]	**N**otário, ca**n**ário
<o,O>	[ɔ]	P**o**rta
	[o]	S**o**pa
	[u]	P**o**rtaria
	[w]	Á**gu**a
	[w̃]	limã**o**
<p,P>	[p]	**P**anela, ma**p**a
<r,R>	[ɾ]	Ma**r**oto, a**r**tista
	[ʀ]	**R**odada, hon**r**a
<s,S>	[s]	**S**apato, bal**s**a
	[z]	Ri**s**o, ca**s**a
	[ʃ]	Lápi**s**, a**s**tro
	[ʒ]	A**s**no, e**s**mero
<t,T>	[t]	**T**eatro, cane**t**a
<u,U>	[u]	**U**nha, m**u**la
	[w]	Pa**u**ta, va**u**
<v,V>	[v]	**v**iola, a**v**ião
<x,X>	[ʃ]	**X**aile, ve**x**ame
	[z]	E**x**ame
	[ks]	Tó**x**ico
<z,Z>	[z]	**Z**ebra, a**z**eite

Tabela 3.3: Relação dos símbolos do alfabeto fonético internacional com os símbolos ortográficos (sequências de letras) (adaptada de Mateus et al. (2005))

Sequências de letras	Sons	Exemplos
<ã,an,am,Ã,AN,AM>	[ɐ̃]	F**ã**, b**an**co, c**am**po
<en,em,EN,EM>	[ẽ]	V**en**to, **em**prego
<õ,on,om,Õ,ON,OM>	[õ]	An**õ**es, l**on**tra, c**om**pras
<in,im,IN,IM>	[ĩ]	P**in**tor, **im**posto
<un,um,UN,UM>	[ũ]	M**un**do, **um**bigo

Tabela 3.4: Relação dos símbolos do alfabeto fonético internacional com os símbolos ortográficos (letras com acento) (adaptada de Mateus et al. (2005))

Letras compostas	Sons	Exemplos
<á,à,Á,À>	[a]	Armár**io**, **à**
<é,É>	[ɛ]	Café
<í,Í>	[i]	Saída
<ó,Ó>	[ɔ]	**Ó**rbita
<ú,Ú>	[u]	Saúde
<â,Â>	[ɐ]	**Â**mago
<ê,Ê>	[e]	Inglês
<ô,Ô>	[o]	Bisavô
<ç,Ç>	[s]	Palha**ç**o

Tabela 3.5: Relação dos símbolos do alfabeto fonético internacional com os símbolos ortográficos (dígrafos) (adaptada de Mateus et al. (2005))

Dígrafos	Sons	Exemplos
<ch,Ch,CH>	[ʃ]	**Ch**apéu
<lh,LH>	[ʎ]	Garga**lh**ada
<nh,NH>	[ɲ]	Gali**nh**a
<qu,Qu,QU>	[k]	**Qu**intal
<rr,RR>	[ʁ]	Ga**rr**afa
<ss,SS>	[s]	A**ss**ador

as cordas vocais é designada por glote, a passagem de ar força as cordas a vibrarem e a produzirem o som vozeado desejado.

O tracto vocal pode ser divido em duas partes, sendo uma a cavidade oral onde se encontram os diversos articuladores (língua, lábios, dentes, palato, etc) e a outra a cavidade nasal. Combinações entre articuladores geram diversos sons, estes são designados de não-vozeados ou surdos. A úvula permite desviar totalmente o ar para uma das cavidades, originando assim um som oral ou nasal.

3.4 Classificação dos Sons

Como referido em 3.3, a conjugação de articuladores dá origem à produção de diferentes tipos de sons, sendo que esta conjugação pode ser dividida em duas partes, uma para as vogais e outra para as consoantes. A geração de vogais é realizada com a vibração das cordas vocais e deixando fluir o ar livremente pelo tracto vocal. Os lábios e a língua são articuladores envolvidos neste processo, conseguindo assim produzir nove vogais orais, cinco vogais nasais, e duas semivogais do Português Europeu Mateus et al. (2005).

Por sua vez e ao contrário das vogais, as consoantes resultam de uma obstrução da passagem do ar ao longo do tracto vocal dando origem à criação de ruído. Combinando os articuladores obtemos quatro tipos de sons, os quais são designados de oclusivos, fricativos, laterais e vibrantes.

As consoantes oclusivas são geradas criando uma obstrução total da passagem do ar e de seguida "explode" libertando o ar retido, estas consoantes encontram-se divididas em mais dois subgrupos. As oclusivas orais são por exemplo [b, p, d, t, g, k] e as nasais são [m, n, ɲ]. A figura 3.2 pretende ilustrar este processo de geração da oclusiva oral [p], assim como os articuladores envolvidos.

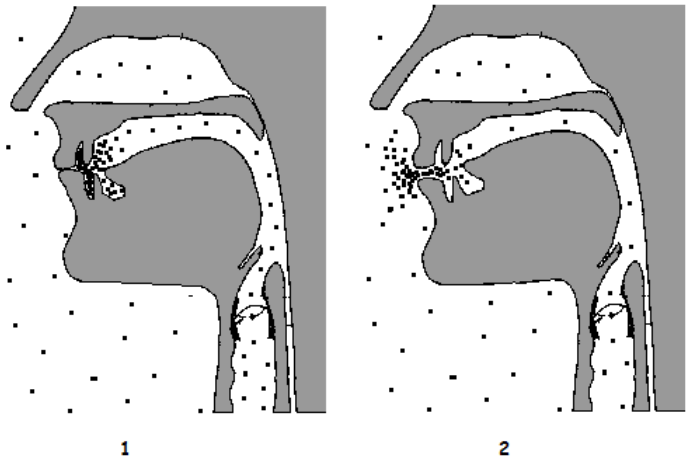

Figura 3.2: Articulação necessária para criar uma consoante oclusiva, neste caso é um [p] (adaptada de http://www.ic.arizona.edu/~lsp/Phonetics/ConsonantsII/Phonetics3d.html, último acesso em 07/05/09).

As consoantes fricativas são produzidas através do estreitamento do tracto vocal, [v, f, z, s, ʒ, ʃ]. A figura 3.3 apresenta o processo e os articuladores para criação da fricativa [ʃ].

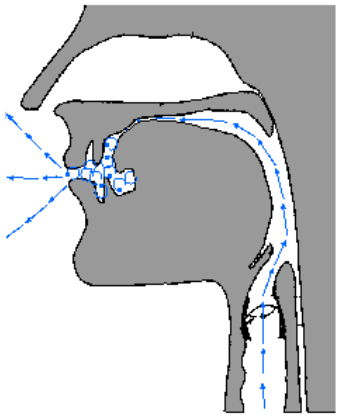

Figura 3.3: Articulação necessária para criar uma consoante fricativa, neste caso é um [ʃ] (adaptada de http://www.ic.arizona.edu/~lsp/Phonetics/ConsonantsII/Phonetics3d.html, último acesso em 07/05/09).

As consoantes laterais realizam-se criando uma obstrução central com a língua, o que faz com que o ar passe pelas laterais da cavidade oral, ou seja, contornando a língua. [l, ʎ] são exemplos deste tipo de sons.

As consoantes vibrantes são fruto de uma obstrução parcial que provoca uma vibração da língua, [ɾ, ʀ] são exemplos.

3.5 Propriedades Físicas

A agitação das moléculas do ar cria zonas com diferentes pressões, estas diferenças são interpretadas pelo nosso sistema de audição resultando em informação para o nosso cérebro processar. O resultado desse processamento é o som. Este pode ser simples (tom puro) ou complexo, sendo que um som simples corresponde a uma sinusóide enquanto um complexo corresponde a soma de diversas sinusóides com frequências, amplitudes e fase diferentes. O pico positivo da sinusóide indica onde ocorre a pressão máxima, enquanto que o negativo corresponde ao de menor pressão. Quanto maior for a distância do pico positivo ao ponto intermédio (corresponde ao zero da sinusóide), maior é a pressão exercida. O intervalo de tempo que uma sinusóide demora a atingir esse ponto novamente é designado de período, que por sua vez é inversamente proporcional à frequência da sinusóide. A distância que um ponto da sinusóide se encontra do início do período é a fase, pode variar entre 0° e 360°.

Os sons da fala são sinais complexos, ou seja, compostos com várias sinusóides com diferentes amplitudes, frequências e fases. Estes sinais complexos podem ser periódicos no caso das vogais, ou aperiódicos no caso das oclusivas, fricativas, laterais e vibrantes. A frequência fundamental (F0) de uma onda sonora é imposta pela vibração das cordas vocais. Devido ao facto do impulso glótico ser mais complexo do que uma sinusóide, F0 vem acompanhada de frequências múltiplas de si própria (nF0, n = 1, 2, 3...). Estas são designadas de harmónicos de F0. Cada indivíduo possui uma F0 que pode variar entre 50 Hz e 500 Hz, sendo que tipicamente para os homens pode ir de 80 Hz e 200 Hz, para as mulheres pode ir de 150 Hz e 350 Hz Mateus et al. (2005).

Os articuladores podem ser modelizados por um filtro, eliminando frequências que não interessam para o som em questão e deixando passar as que interessam com alguma amplificação à mistura. Ao padrão de ressonâncias que se obtém com este efeito de filtragem designa-se por formantes do sinal. A localização das formantes (frequências de ressonância) pode ser modelada de uma forma simplista impondo algumas condições fronteira (mais detalhe em Ferreira et al. (2009)). A seguinte expressão,

$$V(\Omega) = \frac{u(\ell,\Omega)}{u_g(\Omega)} = \frac{1}{\cos\left(\frac{\Omega\ell}{c}\right)}$$

modeliza o tracto vocal relacionando a velocidade volúmica nos lábios e na glote. A partir daqui obtém-se a seguinte expressão que nos permite obter a localização das formantes,

$$\Omega_i \frac{\ell}{c} = (2i-1)\frac{\pi}{2}.$$

Basta substituir o ℓ pelo comprimento de um tracto vocal, o $\boldsymbol{c}$ é a velocidade do som (340 m/s), e o $\boldsymbol{i}$ é a i-ésima formante (i=1, ..., n), para assim se obter a localização das formantes do tracto vocal em questão. As vogais podem ainda apresentar as seguintes características Ferreira et al. (2009): as formantes dependem da área do tracto vocal, a largura de banda de F1 e F2 (as duas formantes de mais baixa frequência) depende das perdas nas paredes do tracto vocal, a largura de banda das formantes de frequência mais elevada depende das perdas por fricção, térmicas, e radiação. Para finalizar existem formantes a cada kHz, excepto para as nasais porque o tracto é mais longo.

Antes de avançar para as ferramentas que são úteis na análise de sinais de audio, interessa explicitar o que é o espectro de um sinal. Esta análise não é mais que analisar o dado sinal no domínio das frequências em vez do domínio do tempo. Por outras palavras, observando a figura 3.4 os gráficos do lado esquerdo representam sinais sinusoidais, sendo que o de cima é uma sinusóide simples o de baixo é uma complexa. Os sinais de fala são complexos, logo este tipo de análise não é muito útil. O analisador espectral que foi referido indirectamente acima (figura 3.4 lado direito) é uma ferramenta importante, visto que nos proporciona uma análise no domínio das frequências num dado instante temporal. Mas a análise em instantes não é muito prática se pretendermos analisar um sinal extenso (uma sequência de palavras por exemplo), sem dúvida que para esta situação uma ferramenta muito mais apropriada será o espectrograma. Os sinais utilizados na figura 3.4 são estacionários, ou seja, são periódicos e seu espectro é igual em qualquer instante do tempo. Em contra partida, os sinais da fala não possuem estas características tão simplistas. Devido a isso teríamos de analisar a evolução espectral em cada instante, sendo que para analisar sinais de fala não é muito apropriado visto que seria necessário realizar muitas visualizações do espectro. Então, se em vez de olhar individualmente para estas análises instantâneas se juntarem todas numa única representação obtemos a ferramenta mais importante para a nossa análise. O espectrograma é apresentado em duas dimensões mas pode ser considerado de três porque mostra a evolução das frequências ao longo do tempo e a sua intensidade. Ou seja, no eixo dos xx´ s temos o tempo

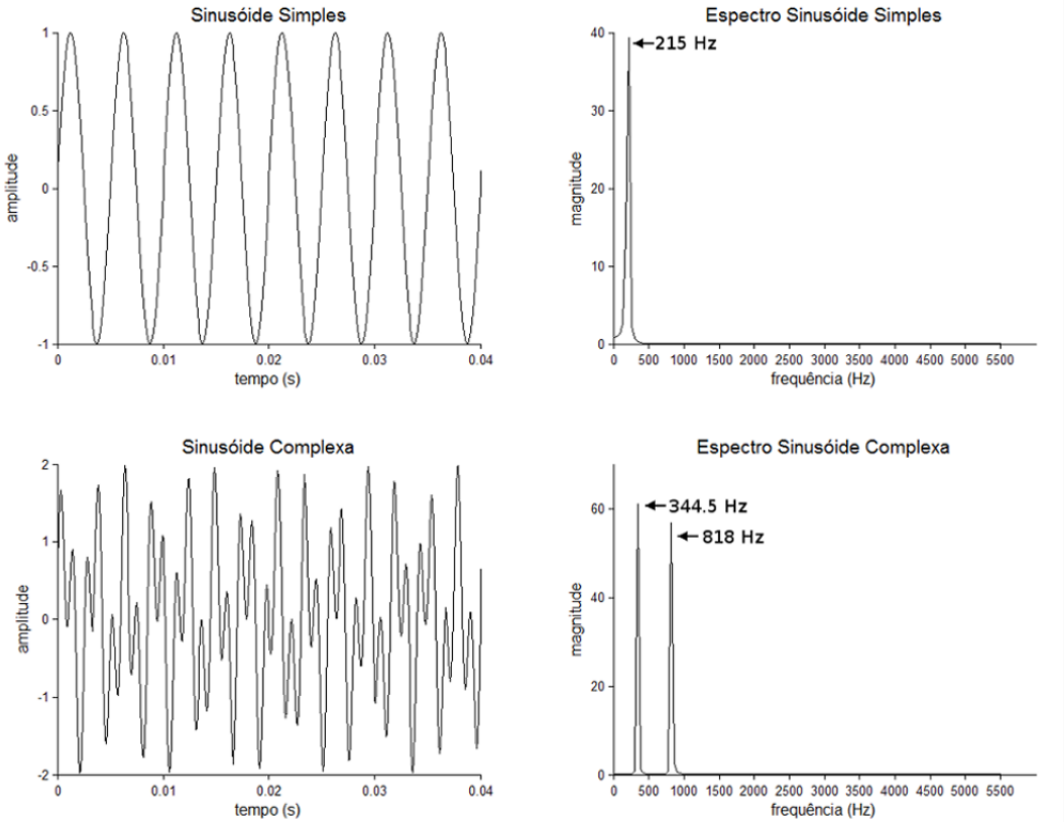

Figura 3.4: O lado esquerdo representa a forma da onda no domínio dos tempos e do lado direito encontra-se o espectro dos sinais.

enquanto que nos yy´ s temos as frequências, os zz´ s são representados pela tonalidade de uma cor (no nosso caso é o preto mas é passível de ser representado com outras cores). A figura 3.5 representa o espectrograma da palavra "animais". Podemos observar a evolução das frequências ao logo do tempo assim como a intensidade (quanto mais escura for a região maior é a intensidade) das mesmas, também podemos destinguir as partes vozeadas das não vozeadas. Na parte vozeada da figura podemos verificar a estrutura dos harmónicos, ou seja, as linhas horizontais mais escuras simbolizam os harmónicos. As formantes por sua vez são identificadas como conjuntos de harmónicos com intensidades consideráveis. Na parte não vozeada verifica-se que não existem harmónicos mas em contrapartida verifica-se a existência de energia acima dos 3.5 kHz e que é uma característica das consoantes.

3.6 Propriedades Segmentais

Cada segmento fonético do Português Europeu contém características próprias, sendo que temos dois grupos importantes, o grupo das vogais e o grupo das consoantes.

Começando pelas vogais, como referido em 3.4, as cordas vogais são a fonte principal dos sons vozeados, estes por sua vez são compostos por formantes nas baixas frequências. Segundo Mateus

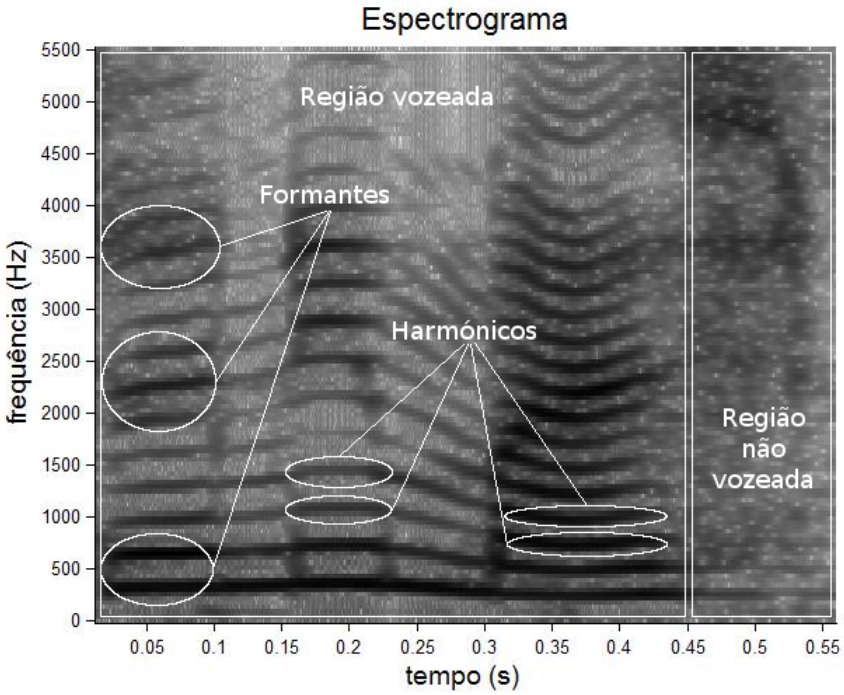

Figura 3.5: Apresentação de um espectrograma da palavra "animais", onde se pode observar a região vozeada e não vozeada, exemplo de formantes e harmónicos.

et al. (2005) as primeiras três formantes são as mais importantes, mas as duas primeiras são suficientes para caracterizar um som. Sendo assim, podemos concluir que quando um sinal apresentar componentes com muita energia nas baixas frequências, trata-se muito possivelmente de um som vozeado.

Por sua vez as consoantes apresentam características que dependem do tipo de articuladores envolvidos, ou seja, por exemplo as oclusivas tem características que as fricativas não possuem e vice-versa. As oclusivas caracterizam-se por passarem por duas etapas na sua formação, inicialmente o ar é retido na cavidade oral e depois é libertado (ver figura 3.2). As palavras constituídas com este tipo de sons apresentam duas variantes, sendo que a primeira apresenta uma ligeira pausa e um aparecimento de energia nas altas frequências, [p, t, k] por exemplo. A segunda variante apresenta um ligeiro aparecimento na zona das vozeadas, e mantendo energia as altas frequências. Também importa referir que as oclusivas nasais apresentam uma característica diferente das suas congéneres orais, isto porque a cavidade nasal é aberta e a "explosão" do ar retido é quase inexistente, logo a sua concentração de energia é basicamente na zona vozeada.

As fricativas caracterizam-se por serem turbulentas, ou seja, com o estreitamento do tracto vocal, a passagem do ar pelos articuladores responsáveis pelo estreitamento, origina uma turbulência que é semelhante a ruído. De acordo com Mateus et al. (2005), o ruído depende de três factores: o local do estreitamento do tracto vocal, ao formato do estreitamento e as propriedades do fluxo do ar. Tal como as oclusivas, algumas fricativas também possuem uma ligeira zona com vozeamento, mas não deixam de ter a maior parte de energia nas altas frequências. Por outro lado, as restantes só apresentam energia significativa nas altas frequências.

As consoantes laterais e vibrantes possuem uma articulação semelhante à das vogais, sendo assim, apresentam componentes na zona vozeada mas não são tão perceptíveis quanto as vogais.

3.7 Propriedades Suprassegmentais

Para além das propriedades segmentais que são relevantes para a identificação de sons, as propriedades suprassegmentais que referiremos em seguida também aportam informação aos sons da fala.

Iniciamos com a duração, que é o tempo intrínseco de articulação de um som, embora dependa do contexto em que é empregue. Mas segundo Mateus et al. (2005) esta propriedade não é importante para o Português Europeu, porque não utiliza diferentes tempos de duração para criar constrastes entre os sons variando o tempo de duração de uma vogal.

Seguindo com o tom e a entoação, em primeiro lugar o tom caracteriza a frequência fundamental (F0) da voz que depende das cordas vocais. A entoação é um conjunto de sucessões de tons, por sua vez é utilizado para criar diferentes contextos ao nível da fala (semântica), por exemplo, a entoação de uma pergunta é diferente da de uma afirmação.

O realce de uma palavra com o aumento das suas propriedades físicas, como a sua intensidade sonora, é um exemplo de proeminência. Esta propriedade permite realçar segmentos sonoros de modo que criem alguma evidência perante o contexto em que se encontra inserido.

O ritmo é importante para o nosso cérebro poder perceber algumas propriedades físicas dos sons de fala e assim poder associá-los a padrões rítmicos.

Capítulo 4

Estado da Arte

4.1 Introdução

De alguns anos para cá muitos trabalhos e investigações tem sido realizados nesta área para se conseguir superar o problema causado pela perda de audição às altas frequências (Power, 1989; Reed et al., 1983; Robinson et al., 2007; Fraga et al., 2008). Alguns trabalhos realizados com amplificação das altas frequências, *vocoders*, *zero-crossing-rate division*, e *slow playback*, translação de frequências, compressão de frequências, e translação/compressão são algumas das técnicas que já foram amplamente testadas. Os resultados obtidos por este sistemas mostram que nem todos têm potencial para se continuar a apostar neles, visto que introduzem distorções que degradam a qualidade do som e alteram características importantes. Alterando características como o *pitch*, a envolvente espectral, a duração dos elementos segmentais entre outros, dificulta ainda mais a tarefa de percepção do discurso para indivíduos com perda profunda a partir de 2 kHz.

Um pilar fundamental para superar este problema, será sem dúvida tirar partido do facto dos indivíduos manterem as qualidades auditivas a baixas frequências quase inalteradas. Com base neste propósito já existem trabalhos com resultados que indicam que se está a caminhar no bom sentido (Fraga et al., 2008; Paarmann, 2006; Simpson et al., 2005).

4.2 Translação de Frequências

Uma das primeiras técnicas usadas para tirar o máximo partido da capacidade de percepção dos sons de áudio às baixas frequências, foi a técnica de translação das frequências. Power (1989) decidiu realizar um melhoramento sobre um trabalho de Posen et al.(1984). Os testes realizados a esse trabalho com sons constituídos por C/a/ [1], e por /b/V/t/ [2] revelaram que quase 90% dos sons

[1] C/a/ - consoante e vogal *a*

[2] /b/V/t/ - consoante *b*, vogal e consoante *t*

foram reconhecidos. Mas quando utilizado com expressões maiores a compreensão tornou-se muito difícil.

A técnica inicial consistia num sistema passa baixo até os 0.8 kHz ao qual eram adicionados sinais de ruído quando apareciam oclusivas. O intervalo de tempo das oclusivas é estimado em tempo real medindo a distribuição de energia do sinal de fala. O ruído nas quatro bandas (0.35-0.8 kHz) de baixas frequências tem energia proporcional à das quatro bandas de altas frequências, e só é gerado quando a razão da energia acima de 1.4 kHz e abaixo desta, for maior que -0.8 dB. As alterações introduzidas consistem em permitir a geração de ruído durante a oclusiva e criar um ruído específico para cada tipo de oclusiva. Também foi necessário introduzir sinais artificiais nos segmentos que contêm consoantes nasais, pois os sons /m/ e /n/ foram causa de confusão no primeiro trabalho.

Sons formados por CV[3] foram aqui utilizados para testar as alterações utilizadas que resultaram em ligeiros melhoramentos no reconhecimento de sons nasais de semivogais, africados e plosivas vozeadas. Em contrapartida, não resultou em melhoramentos de plosivas vozeadas de nasais, em fricativas vozeadas e não vozeadas. As alterações introduzidas sobre a primeira técnica revelaram uma melhoria muito ligeira, e a diferença de resultados talvez derive de um *tradeoff* entre reduzir a variabilidade (é útil durante o reconhecimento de sílabas) e manter a estrutura suprasegmental (ajuda na identificação de expressões) dos sinais de voz.

O trabalho de Robinson et al. (2007) segue na base de outros que realizaram translação de frequências para ajudar a superar a perda auditiva nas altas frequências. Este difere de todos os outros por duas razões. A primeira é que utiliza um algoritmo baseado em FFT com translação condicional mas sem *slow playback*[4], e sem compressão de frequências. A segunda é que todos os sujeitos tem diagnosticada uma perda auditiva as altas frequências (*high frequency dead region*)[5] com uma frequência (f_e) bem definida, pois esta estabelece a fronteira entre a zona das frequências que não podem ser alteradas (baixas frequências) e a zona na qual se colocam as frequências transladadas. O algoritmo é adaptado a cada sujeito com base na sua DR[6], e segundo French and Steinberg (1974) a informação do discurso não se encontra distribuída uniformemente na gama de frequências, devido a isso a informação nas bandas adjacentes é parcialmente redundante e o sistema tem isto em conta.

O algoritmo de translação tem isso em conta, usando como banda de destino de f_e até $1.7f_e$, sendo a banda de altas frequências $2f_e$ até $2.7f_e$ transladada para a banda de destino. A figura 4.1 ilustra as bandas descritas anteriormente. O algoritmo (ver figura 4.2) foi implementado em Matlab com amostras de 16 kHz que depois são decompostas em janelas de 128 amostras e multiplicadas por outra janela sinusoidal (*half-sine*). Ao resultado obtido é aplicado uma FFT para realizar uma conversão do domínio dos tempos para o domínio das frequências, assim obtêm-se passos de 0.125 kHz do espectro da frequência de entrada. Extraindo assim a magnitude e a fase de

[3] CV - Consoante Vogal

[4] slow playback - técnica utilizada por outro trabalho, referenciado pelos autores, esta técnica consiste em reproduzir os sons de uma forma mais lenta que a original

[5] nome dado pelos autores

[6] DR - região morta da cóclea

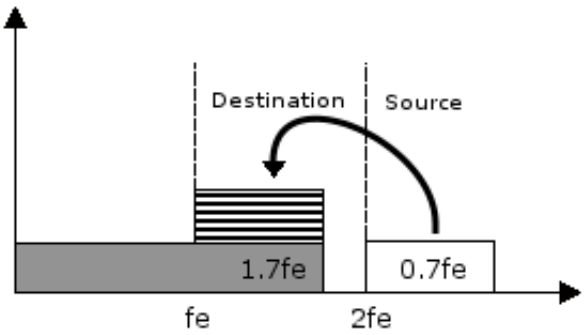

Figura 4.1: Esquema das bandas utilizadas pelo algoritmo (adaptada de Robinson et al. (2007))

cada amostra espectral, esta informação é importante para tomar a decisão de realizar translação ou não. Ao elevar ao quadrado a magnitude obtém-se a potência/bin, é realizado um somatório dessas potências e assim que se atinge os 1.25 kHz é definida a potência para as baixas frequências. Para as altas o somatório é realizado a partir da baixa até os 8 kHz. Se a relação entre a potência as altas e baixas frequências,

$$\frac{P_{HF}}{P_{LF}} > 0,1$$

onde P_{HF} é a potência altas frequências, P_{LF} é a potência baixas frequências, então é realizada a translação. O sinal de saída passa por um filtro passa baixo, tenha ou não existido translação,

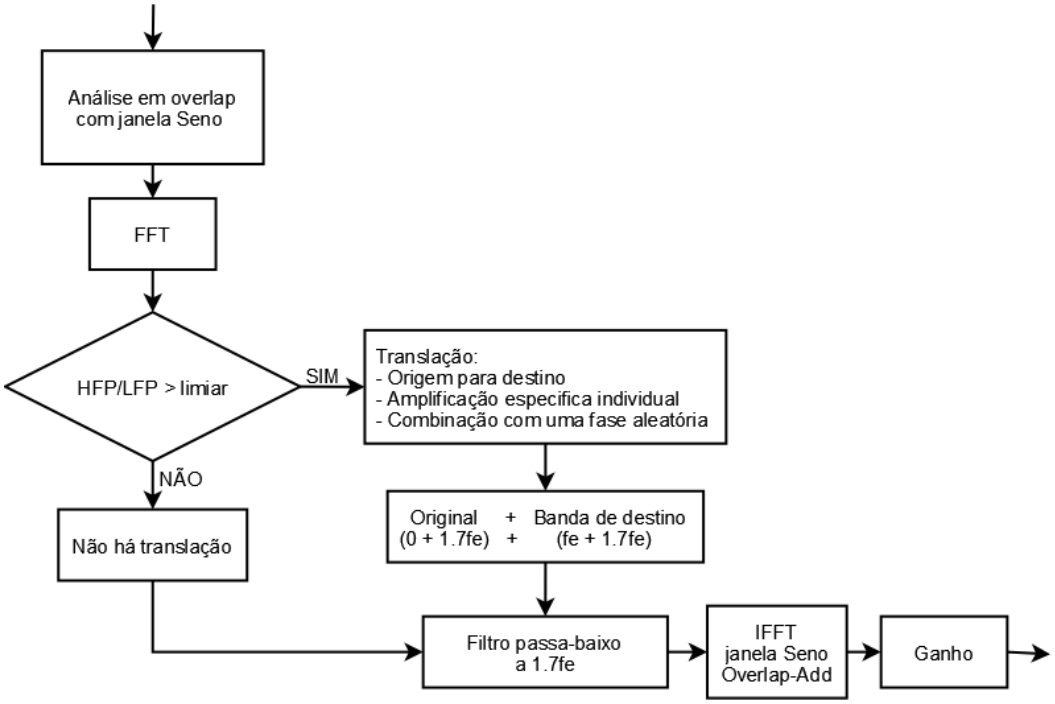

Figura 4.2: Diagrama de blocos do algoritmo (adaptada de Robinson et al. (2007)).

para eliminar componentes de ruído ou elementos de altas frequências que possam ainda existir.

Para colocar o sinal na saída é preciso realizar a conversão do domínio das frequências para o dos tempos, para isso é utilizada a mesma técnica que no início, mas agora utiliza uma IFFT em vez da FFT. É utilizado o método *overlap-and-add*, em que são utilizadas 64 amostras na parte de *overlap* e todo multiplicado por um ganho específico. O ganho inicialmente utilizado era de 4, pois aos ouvidos dos investigadores o som reproduzido era de boa percepção, mas quando foi testado por indivíduos com deficiência auditiva rapidamente se concluiu que o som era desconfortável. Devido a isso, foi introduzido um novo ganho que é obtido utilizando a formula de Cambridge,

$$IG(f) = HL(f) \times 0.48 + INT(f)$$

onde IG(f) é o ganho desejado a cada frequência, HL(f) é a perda auditiva do indivíduo, INT(f) é um limiar dependente da frequência.

Com este ganho consegue-se obter uma sonoridade praticamente igual para todas as frequências. Os resultados sobre os sons constituídos por VCV[7] revelaram um melhoramento no reconhecimento de sons africados, alguns sons fricativos melhoraram significativamente mas em contrapartida gerou novos tipos de confusão entre os sons. Por exemplo confusões entre /ʧ/ e /t/, /ʤ/ e /d/ foram significativamente reduzidas, mas aumentou para /ʃ/ e /s/. Assim fazendo com que a sonoridade de /s/ e /f/ parecerem /ʃ/. Por último, utilizando uma palavra singular e o seu plural mais de metade dos indivíduos conseguiram distinguir a diferença.

4.3 Compressão de Frequências

Tendo em vista os fracos resultados obtidos pelos trabalhos anteriores com translação, ou mesmo, com as primeiras tentativas de melhorar a qualidade do som para os indivíduos que sofrem de perda auditiva na região das altas frequências, Reed et al. (1983) decidiram partir para a técnica de compressão das frequências, sendo dos primeiros a utilizá-la. Aqui realizaram um trabalho comparativo de discriminação de consoantes com rebaixamento nas frequências e um filtro passa baixo para simular a deficiência. A técnica utilizada para comprimir as frequências altas segue quatro passos, o primeiro consiste em segmentar o sinal em intervalos temporários, o segundo é processar os segmentos com uma operação de *warping*, o terceiro consiste em aumentar no tempo os segmentos resultantes do segundo passo e por fim colocar na saída os segmentos processados. A frequência fundamental é preservada porque o espaçamento entre os períodos do tom é salvaguardado no processamento realizado. A frequência de saída (ver figura 4.3) é dada pela seguinte expressão:

$$f_{out} = \frac{f_s}{K\pi} \arctan\left[\frac{1+a}{1-a} \tan\left(\frac{\pi f_{in}}{f_s}\right)\right]$$

onde K é o factor de rebaixamento de frequências, a é factor de compressão. Na realização dos

[7] Vogal Consoante Vogal

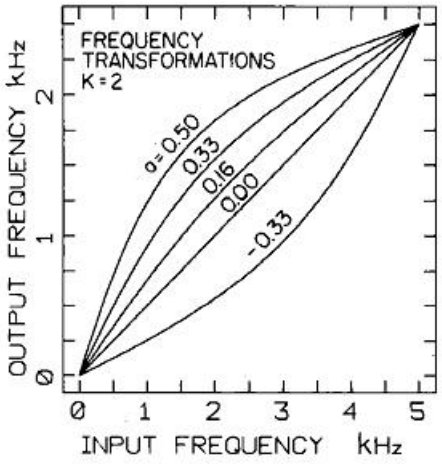

Figura 4.3: Mapeamento da frequência de entrada para a frequência de saída (adaptada de Reed et al. (1983)).

testes foram utilizados sons compostos por CV e vários valores para os parâmetros K e a. Os resultados obtidos dos testes anteriores mostraram que no reconhecimento do contraste entre semivogais e nasais, o filtro passa baixo apresentou melhores resultados que qualquer tipo de rebaixamento de frequências (variações do K e do a). O reconhecimento de plosivas foi similar entre os dois, enquanto que nas fricativas o rebaixamento surgiu com um ligeiro melhoramento. Utilizando uma frequência de corte de 2.5 kHz obtiveram-se melhores resultados do que com 1.25 kHz, isto deve-se ao facto de que a informação da segunda formante é preservada. Com factores de 2 e 3 para a largura de banda da compressão, degradam a inteligibilidade do som. Sendo assim conclui-se que tanto o filtro passa baixo como o rebaixamento de frequências apresentaram mais ou menos os mesmos resultados. Mas este tipo de rebaixamento de frequências obteve um resultado positivo, pois consegue manter praticamente intacta a parte de baixa frequência do sinal.

Os investigadores deste trabalho que acabamos de ver, decidiram realizar um novo estudo Reed et al. (1985) para alargar os testes realizados em Reed et al. (1983). Desta vez realizaram rebaixamento de frequências e amplificação linear através do mesmo tipo de sons. Os resultados obtidos são semelhantes ao anteriores, pois os indivíduos apresentaram uma melhoria na distinção das fricativas mas em contrapartida nas semivogais pioraram. Sendo assim, os indivíduos deste segundo trabalho não apresentaram melhorias satisfatórias com o rebaixamento de frequências em relação a amplificação linear.

Dando seguimento à técnica utilizada por Reed et al. (1983), Muñoz et al. (1999) decidiram efectuar modificações baseados na hipótese que o factor de compressão utilizado para os sons fricativos e africados degradava o reconhecimento de sons nasais, semivogais e vogais. Sendo assim decidiram incluir um algoritmo que detecta a energia as altas frequências. O algoritmo detecta os sons fricativos e africados e decide se deve processar o sinal. A segunda alteração consiste em determinar a distância entre os picos do sinal, e assim poder manter a relação no sinal processado. O resultado deste novo processamento manteve o *pitch* fundamental e o padrão temporal com boa qualidade de som. Quanto aos resultados indicam que as alterações provocaram

uma melhoria na percepção dos sons fricativos, plosivos e africados. Em contrapartida, como vimos nas alterações efectuadas, não é realizado qualquer tipo de processamento sobre o resto dos sons, por isso a percepção de vogais e sons nasais mantiveram-se iguais ao trabalho anterior.

Verificando que a compressão de frequências é uma técnica que vem apresentando alguns resultados positivos, Turner e Hurtig (1999) decidiram basear o seu trabalho nesta técnica. Com recurso à função *pitch shift* de um software de tratamento de áudio, foram realizadas compressões com factores 0.9 até 0.5. O algoritmo da função realiza a compressão do sinal alterando minimamente a sua envolvente temporal e duração. Para determinar se a técnica usada apresenta algum tipo de beneficio para os indivíduos que participaram nos testes, realizaram-se testes com palavras constituídas por CV. Durante a realização dos testes foram utilizados três indivíduos com audição normal que foram submetidos a diversos factores de compressão, desde 0.9 até aos 0.7. O som reproduzido era perceptível mas abaixo do limite inferior, deixou de o ser. Sendo assim, se um indivíduo com audição normal deixa de reconhecer o som então é um bom indicador do factor máximo que pode ser usado para os indivíduos com deficiência. Comparando o resto dos testes com esta técnica e com um aparelho comercial de amplificação linear, verifica-se um ligeiro melhoramento na percepção dos sons, embora este resultado varie entre indivíduos e os factores de compressão também. Outra constatação é que os indivíduos que apresentam níveis elevados de surdez obtiveram melhores resultados que os que têm um nível menos elevado. Por fim esta técnica não consegue incrementar inteligibilidade de todos os fonemas, em particular os fricativos.

No seguimento da técnica de compressão de frequências, Simpson et al. (2005) criaram uma estrutura recorrendo a dois aparelhos convencionais e um processador de sinal (ver figura 4.4). Os

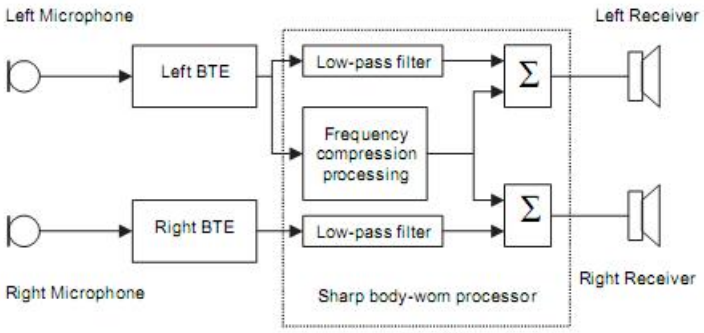

Figura 4.4: Estrutura do hardware utilizado (adaptada de Simpson et al. (2005))

sinais de entrada são separados em dois canais, um canal passa baixo onde o sinal é colocado na saída sem processamento e um canal passa alto no qual é executado o algoritmo de compressão de frequências não-linear. As amostras provenientes do canal passa alto são processadas em blocos de 256 amostras, que passam por uma janela temporal criada a partir de um produto de uma janela de Hamming e de um seno cardinal (sinc). O resultado produzido pela janela é processado por uma FFT de 128 amostras, a qual fornece informação sobre a magnitude do sinal e a fase. Com base nessa informação é colocada na saída a amplitude e a fase que é usada para modular um bancos de osciladores sinusoidais. Os primeiros 24 bins das FFTs acima da frequência de corte

são associados aos 24 osciladores, a frequência é actualizada utilizando a seguinte formula,

$$F_{out} = \begin{cases} F_{in}, & F_{in} < F_{LPF} \\ F_{LPF}^{1-p} \times F_{in}^{p}, & F_{in} \geq F_{LPF} \end{cases}$$

onde F_{LPF} é a frequência de corte, p é o factor de compressão. Palavras monossilábicas compostas

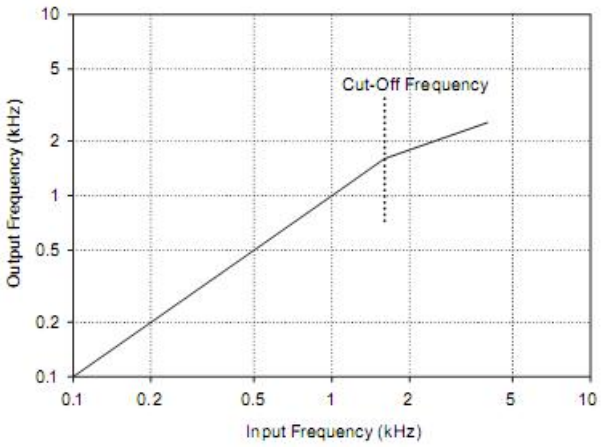

Figura 4.5: Curva característica de compressão (adaptada de Simpson et al. (2005))

por CNC [8] foram utilizadas para realizar os testes, resultando num melhoramento nos sons fricativos. Também se verificou que não há diferenças significativas na transmissão para qualquer tipo de articulação entre o aparelho convencional e este. As sonoridades constituídas basicamente com frequências baixas, não são afectadas pela compressão.

No sistema referido acima existem dois parâmetros que podem ser ajustados, o ponto onde se inicia a compressão e o segundo é o factor de compressão. Então Simpson et al. (2006) decidiram ajustar os paramêtros de varias maneiras para se determinar qual seria a melhor combinação. Foram escolhidas as frequências de 1.25 kHz, 1.6 kHz e com um factor de compressão de 2:1. Foram realizados os seguintes testes:

- Reconhecimento de palavras em ambiente silencioso com palavras compostas por CVC[9];
- Reconhecimento de consoantes em ambiente silencioso com expressões compostas por VCV[10];
- Reconhecimento de frases em ambiente ruidoso.

Os testes foram realizados pelo sistema desenvolvido e por uma protese auditiva existente no mercado.

O primeiro e o segundo teste não revelaram em média nenhuma melhoria significante entre os dois aparelhos, embora um indivíduo tenha apresentado melhoria com a compressão de frequências. O terceiro teste apresenta algumas melhorias na maioria dos indivíduos. Este tipo de

[8] Consoante, vogal Nuclear e Consoante
[9] Consoante Vogal Consoante
[10] Vogal Consoante Vogal

compressão conseguiu obter melhorias em alguns fonemas fricativos tais como /sh/ e /j/, em contrapartida piora para fonemas como /g/ que é confundido com o /z/, e o /s/ também piorou. Os sons fricativos /sh/, /z/ e /v/ também apresentaram melhorias. É possível que esta compressão de frequências tenha exibido alguma melhoria em algum tipo de sons, devido a isso resultando no desfavorecimento de outros. Os autores acreditam que talvez com mais treino por parte dos indivíduos o sistema apresentado melhore.

4.4 Translação e Compressão de Frequências

Até agora vimos as técnicas de translação e compressão de frequências em separado. Como apresentaram alguns resultados positivos, Paarmann (2006) pretendeu retirar partido do melhor de cada técnica. A técnica implementada consiste em criar dois canais, um com filtro passa baixo para os sinais de baixas frequências e que não precisam de processamento (ver figura 4.6). O segundo canal implementa um algoritmo para processar os sinais de altas frequências. O algoritmo implementado consiste em recolher amostras e passá-las por uma janela de Hanning, de seguida aplicar a compressão e translação e passar o resultado por um filtro passa banda. O resultado final dos dois canais são somados para criar uma única saída. A compressão e translação das frequências

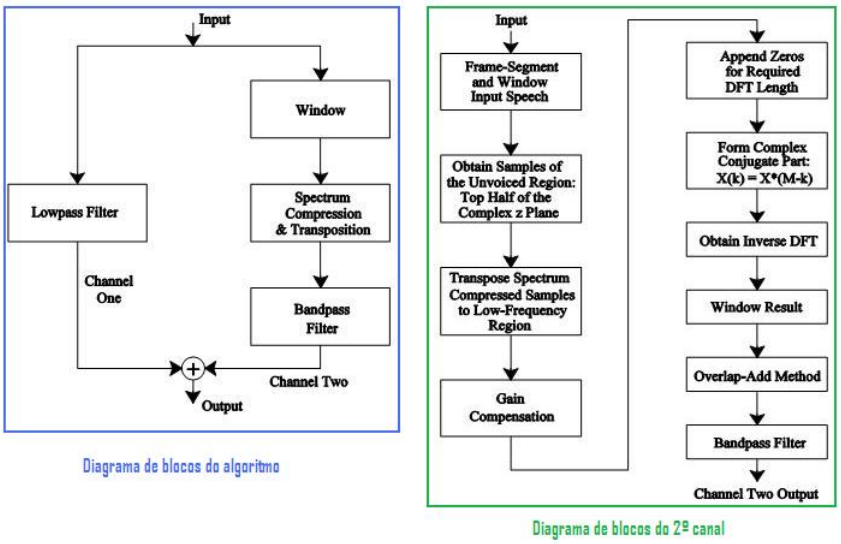

Figura 4.6: Diagrama de blocos do algoritmo (adaptada de Paarmann (2006))

é realizada com recurso ao algoritmo Paarmann e Guiher (1989) que utiliza a transformada Chirp-Z modificada, resultando assim num mapeamento das altas frequências [4.4, 10.5]kHz para quatro bandas e meia de baixa frequência [0.3, 0.84]kHz (ver figura 4.7). Foram efectuados testes com palavras constituídas por CV ou VC[11], mostraram que com esta técnica o reconhecimento das consoantes foi bom. Segundo o autor, se os indivíduos forem treinados os resultados podem melhorar ainda mais.

[11] Vogal Consoante

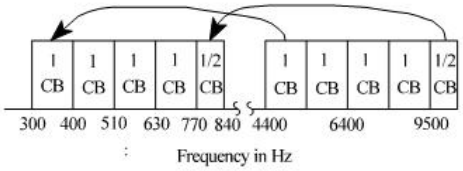

Figura 4.7: Mapeamento das bandas críticas das altas frequências para as baixas (adaptada de Paarmann (2006))

Muitos dos trabalhos realizados até ao momento revelaram que devido ao aumento das confusões entre os sons fricativos, resulta um agravamento do reconhecimento na identificação de outros. Baseados neste facto, Fraga et al. (2008) decidiram partir para a implementação de um algoritmo de compressão e translação que só se aplique às consoantes fricativas e assim tentar evitar o aumento das confusões. Outro aspecto que não foi tido em conta em trabalhos anteriores, é que não é realizada a compressão tendo em conta a forma espectral das fricativas. Com base nestes factos foi implementada uma função linear por partes para realizar a compressão e translação tendo em conta o espectro médio das fricativas brasileiras. Esta média que depois servirá de referência no algoritmo, foi obtida com recurso a uma base de dados com sílabas CV.

O algoritmo utiliza na análise tramas de 50 ms com sobreposição de 75 % que passam por uma janela de Hamming. Em seguida é executada um FFT de 2048 pontos para realizar a classificação do som (vozeado ou não vozeado) através do cálculo de *spectral flatness measure*. O resultado é comparado com a referência e dependendo da condição é realizada ou não a compressão/translação. Na reconstrução do sinal é realizada a ordem inversa da análise, ou seja, é aplicada uma IFFT ao sinal processado, e de seguida e multiplicado pela janela e finalizando com um *overlap-and-add* de 75 %. A figura 4.8 representa a curva da compressão por partes linear,

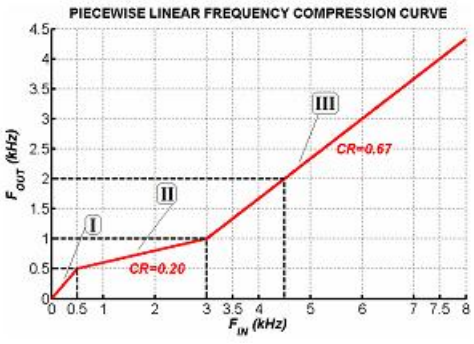

Figura 4.8: Curva de compressão linear por partes (adaptada de Fraga et al. (2008))

esta é dividida em três partes:

I - região de 0 a 0.5 kHz, mantém-se igual para manter a percepção da tonalidade das fricativas vozeadas.

II - região de 0,5 a 3 kHz, nesta parte do espectro apenas as fricativas /ʃ/ e /ʒ/ possuem sinais para identificação dos fonemas. Aqui é aplicada uma compressão forte (0.2) porque os sinais

continuam até os 6.5 kHz e não existe problemas com outros tipos de sons.

III - região de 3 a 8 kHz aqui é aplicada uma compressão mais moderada (0.67) uma vez que se pretende manter informação original do discurso, porque é nesta região que se encontra. A região pode ser dividida em duas partes, a primeira de 3 a 4.5 kHz é sem dúvida a parte mais importante pois a maioria dos deficientes auditivos tem uma frequência de corte entre 1 e 2 kHz. A segunda é de 4.5 até a frequência de Nyquist.

Na realização dos testes foram utilizadas palavras CV e CVC, utilizando combinações das seis fricativas (/s/,/z/,/f/,/v/,/ʃ/, /ʒ/) usadas no Brasil. Os testes foram realizados simulando a deficiência auditiva através de filtragem passa baixo em indivíduos sem deficiência, com duas frequências de corte (1.5 kHz e 2 kHz). Para a frequência de corte de 1.5 kHz os resultados obtidos sobre o orador masculino foram significantes, mas para os 2 kHz o desempenho foi melhor sobre o orador feminino. Com o orador masculino ocorreram confusões com o grupo /s/ e /z/ e o grupo /ʃ/ e /ʒ/, uma vez que o tracto vocal masculino é maior do que o femenino, o que se repercute nas frequências de ressonância fazendo com que sejam mais baixas.

4.5 Conclusão

Depois da análise realizada às soluções existentes, passamos à verificação dos pontos fortes e fracos de cada uma das técnicas de modo a concluir sobre as soluções mais promissoras a serem exploradas no nosso trabalho.

Translação

O trabalho Power (1989) não apresenta quase nenhuma melhoria sobre o seu antecessor, sendo assim, não é uma solução interessante para nós. Isto porque esta não apresentou resultados satisfatórios quando confrontada com sequências de palavras.

Embora o seguinte trabalho Robinson et al. (2007) apresente um bom resultado quando confrontado com uma palavra no singular e com o seu plural, em contrapartida cria muitas confusões entre os sons, e providenciou poucos resultados práticos satisfatórios para ajudar na compreensão das consoantes. O seu teste foi realizado *offline*, criando assim mais uma incerteza quanto a viabilidade desta solução, visto que é nossa ambição desenvolver uma aplicação em tempo real.

Depois de analisar as soluções propostas, pensamos que nenhuma delas seja totalmente viável para criar uma base para o trabalho a desenvolver, embora algum aspecto positivo venha a ser tido em conta.

Compressão

Analisando agora os trabalhos realizados com recurso a esta técnica, e começando por Reed et al. (1983), o reconhecimento dos sons /ʃ/ e /ʒ/ beneficiaram com esta técnica. Este sistema tem a característica de alterar minimamente o espaço das baixas frequências, desta maneira os sons

desta banda não são alterados quando existe processamento das componentes de altas frequências. Mas comparando a compressão com o filtro passa baixo, verificou-se que aproximadamente os resultados foram parecidos.

Os resultados de Turner e Hurtig (1999) mostram que o som resultante de alguns fonemas tais como sons fricativos /s/ e /f/, parecem não naturais. O factor de compressão não pode ser inferior a 0.7 porque senão a reprodução do som deixa de ser perceptível. Esta solução foi realizada com recurso a uma função de um software comercial de áudio.

Para finalizar a análise sobre a compressão, o trabalho Simpson et al. (2005) foi implementado em tempo real, com esta estrutura experimental os indivíduos recebiam mais informação de características de consoantes de altas frequências. Os resultados obtidos foram realizados entre um aparelho comercial e este experimental.

Com base nas análises realizadas a este tipo de técnica, decidiu-se que Simpson et al. (2005) poderá ser uma base de partida. Pois foi realizada em tempo real e apresentou alguns resultados interessantes no reconhecimento de fricativas. Quanto a Reed et al. (1983), como não apresenta resultados muito melhores que a filtragem passa baixo, decidiu-se não optar por esta. Como Turner e Hurtig (1999) foi implementada recorrendo a um software especifico de áudio, é reduzido o seu interesse prático para o nosso trabalho.

Translação e compressão

Relativamente a técnica de translação e compressão, o trabalho de Paarmann (2006) apresentou resultados bons no que diz respeito ao reconhecimento de consoantes, outro aspecto a ter em conta é que tem um atraso de 30.7ms.

Quanto à solução de Fraga et al. (2008), também apresentou bons resultados no reconhecimento de fricativas, pois o seu processamento está direccionado para o processamento de consoantes compostas de altas frequências, embora apresente uma ligeira confusão entre o grupo de /s/ e /z/, e o grupo de /ʃ/ e /ʒ/.

Estas duas soluções apresentam resultados promissores, pelo que podem ser uma boa base para o nosso trabalho.

Conclusão Final

Com base na análise sobre as soluções existentes, de entre todas decidimos optar pelas seguintes soluções como base para o nosso trabalho Fraga et al. (2008), Paarmann (2006) e Simpson et al. (2005) que contemplam os seguintes princípios funcionais principais:

1. não alteram a zona das baixas frequências,
2. só actuam sobre os sons não vozeados,
3. estrutura simples capaz de ser implementada em tempo real.

Capítulo 5

Implementação em MatLab

Este capítulo descreve o trabalho realizado, incluindo a apresentação dos diferentes algoritmos implementados na ferramenta Matlab, a simulação de cada um deles no mesmo ambiente e os resultados obtidos.

5.1 Algoritmos

Esta secção apresenta detalhadamente os quatro algoritmos realizados que são os seguintes:

1. Compressão/Translação: o primeiro algoritmo implementado corresponde a realizar uma compressão da região das frequências acima dos 2 kHz. Através da selecção dos bins da FFT é realizada uma translação destes para a região dos 1.5 kHz aos 2 kHz.

2. Frequency Warping/Translação: o segundo algoritmo corresponde a realizar uma compressão não-linear com recurso a um sistema basedo em filtros passa-tudo, da região das frequências acima dos 2 kHz.

3. Translação ou Compressão: o terceiro algoritmo realiza uma translação, se existir um único pico de energia relevante na zona acima dos 2 kHz, pelo contrário se existir mais que um é realizada uma compressão.

4. Compressão/Translação e Síntese de Harmónicas: o último algoritmo é igual no tratamento das altas frequências ao primeiro algoritmo, com a particularidade de detectar quando uma vogal /i/ ou uma vogal /e/ aparece, e sintetizar na região das baixas frequências sinusóides com características destas vogais.

O código dos algoritmos encontram-se nos anexos A, B, C, e D.

5.1.1 Compressão e Translação

A base deste algoritmo é o trabalho realizado por (Fraga et al., 2008), no qual é realizada uma compressão /translação recorrendo a uma função linear por partes (ver figura 5.1), a função é constituída por três partes lineares. A região I (0 a 500 Hz) não exerce qualquer tipo de alteração no sinal quando surgir alguma fricativa vozeada. Por sua vez a região II (500 Hz a 3 kHz) realiza uma compressão forte porque esta região do espectro não tem segundo os autores informação relevante. Por fim a região III (3 kHz a 8 kHz) efectua uma compressão moderada porque esta zona contém informação espectral das fricativas relevante. O controlo que determina se há processamento do

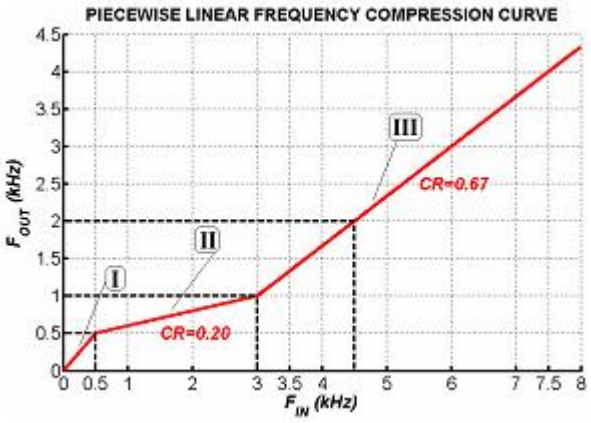

Figura 5.1: Curva de compressão linear por partes (adaptada de Fraga et al. (2008))

sinal ou não, é efectuado comparando o espectro obtido com recurso à *spectral flatness measure* da trama actual, com o espectro médio das fricativas brasileiras (o espectro médio foi calculado com STFT e com uma base de dados).

O nosso algoritmo realiza uma compressão/translação da região dos 2 kHz aos 5.5 kHz para a região dos 1.5 kHz aos 2 kHz. A figura 5.2 apresenta um diagrama de blocos com a implementação do nosso algoritmo, podemos observar que o sistema apresenta uma estrutura com dois canais, um canal passa-baixo (canal 1) e o outro para o processamento (canal 2). A análise do sinal de entrada é realizada em tramas de 256 elementos com uma sobreposição de 50%, e multiplicadas por uma janela seno, sendo ainda o resultado entregue aos dois canais no domínio dos tempos. O canal 1 por sua vez, realiza uma filtragem do tipo passa-baixo com uma frequência de corte de 2 kHz, evitando assim que as altas frequências das partes vozeadas possam criar alguma ambiguidade ou desconforto ao passar pela cóclea danificada. Após a filtragem, o sinal é reconstruído utilizando o método *overlap add* com uma sobreposição de 50%. O canal 2 apresenta uma estrutura mais complexa que o canal 1. O sinal que lhe é entregue no domínio dos tempos é passado para o domínio das frequências através de uma FFT de 256 pontos. É preciso calcular a potência da

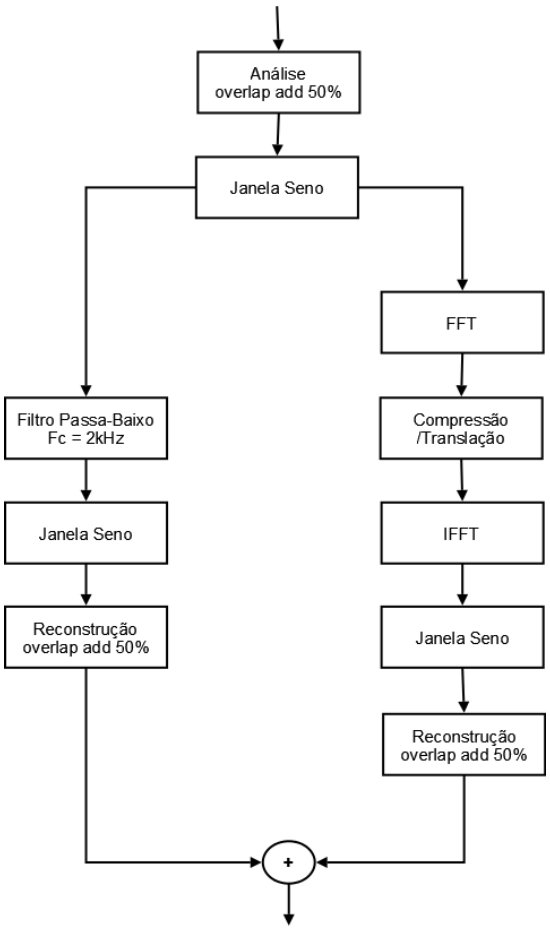

Figura 5.2: Diagrama de blocos do nosso algoritmo de compressão/translação.

região de 0 a 2 kHz (PLF) e na região de 2 a 5.5 kHz (PHF),

$$\begin{aligned} PLF &= \frac{\sum_{n=0}^{46} |X(n)|^2}{47}, \\ PHF &= \frac{\sum_{n=46}^{N/2} |X(n)|^2}{\frac{N}{2} - 46}. \end{aligned} \tag{5.1}$$

Se

$$\frac{PHF}{PLF} > 1.42, \tag{5.2}$$

então significa que estamos perante um som não-vozeado (o espectro das consoantes apresenta maior energia as altas frequências ver 3.4), sendo assim é preciso efectuar processamento. Caso a condição 5.2 não seja satisfeita, não é efectuado nenhum processamento e o sinal passa para o bloco da IFFT. A figura 5.3 ilustra o diagrama de blocos do bloco compressão da figura 5.2, a compressão é realizada pela seguinte equação 5.3

$$f_{out} = 0.125 f_{in} + 1312.5, \tag{5.3}$$

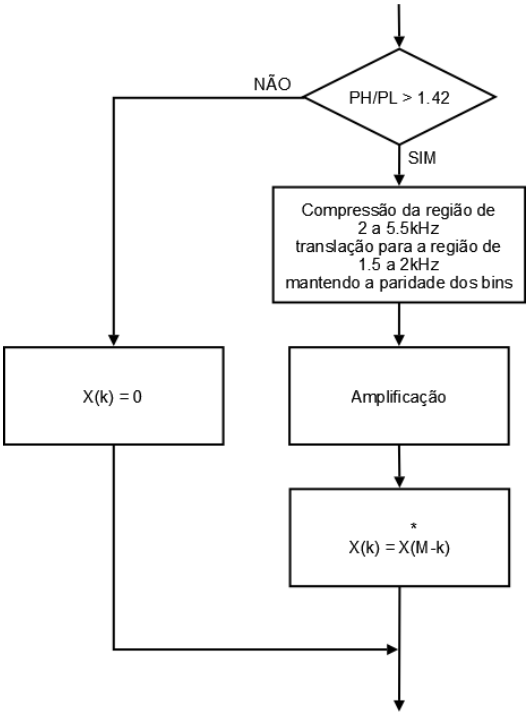

Figura 5.3: Diagrama de blocos do bloco compressão/translação.

onde f_{out} é a nova frequência e f_{in} é a frequência que desejamos mapear. O processo de translação mantém a paridade dos bins, ou seja, por exemplo os bins pares são mapeados para bins pares, isto para evitar distorção na fase do sinal de saída devido à sobreposição de 50%. É necessário realizar uma amplificação pelo facto de o espectro resultante da compressão/translação poder ser mascarado por alguma parte do espectro na região dos 1.5 kHz aos 2 kHz, o ganho é aproximadamente de 1.2. Com este processo só temos metade do espectro, ou seja, as frequências positivas por isso temos de completar a parte das frequências negativas com

$$X(k) = X^*((N-1)-k), \quad k = N-1,\ldots,\frac{N}{2}+1, \tag{5.4}$$

onde N é o número de pontos (256) da FFT. Visto que já foi realizado todo o processamento no domínio das frequências, torna-se necessário realizar a transformada inversa IFFT para voltarmos para o domínio dos tempos, e efectuar o método *overlap add* com 50% de sobreposição. Para finalizar temos de realizar a soma dos dois canais para obtermos o sinal de saída.

5.1.2 Frequency Warping e Translação

O algoritmo de frequency warping/translação utiliza o algoritmo de frequency warping de Ferreira (2007), o mapeamento não-linear é realizado com recurso a um sistema passa-tudo.

Um sistema passa-tudo causal de primeira ordem apresenta a função de transferência seguinte

$$H(z) = \frac{z^{-1} - a^*}{1 - az^{-1}}, \quad |a| < 1. \tag{5.5}$$

A figura 5.4 representa o diagrama zero-polar do sistema passa-tudo. Considerando a complexo,

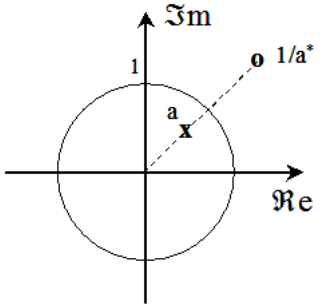

Figura 5.4: Diagrama zero-polar de um sistema passa-tudo de ordem um (adaptada de Ferreira (2007))

de modo a termos uma função de transferência (FT) de coeficientes reais, a deve estar sobre o eixo real. De qualquer modo, a propriedade do sistema baseado em filtros passa-tudo advém do facto do zero da FT ser recíproco conjugado do pólo. Esta configuração de zero e pólo resulta numa magnitude da resposta em frequência constante para todas as frequências. Para verificação considera-se que

$$|H(e^{j\omega})|^2 = H(z)H^*(1/z^*)|_{z=e^{j\omega}} = \frac{z^{-1}-a^*}{1-az^{-1}}\frac{z-a}{1-a^*z} = \frac{1+|a|^2-az^{-1}-a^*z}{1+|a|^2-az^{-1}-a^*z} = 1. \tag{5.6}$$

Admitindo que a é constante e real

$$\begin{aligned} H(e^{j\omega}) &= \frac{\cos(\omega)-j\sin(\omega)-a}{1-a\cos(\omega)+ja\sin(\omega)} \\ &= e^{j\arctan\left(\frac{-\sin(\omega)}{\cos(\omega)-a}\right)-j\arctan\left(\frac{a\sin(\omega)}{1-a\cos(\omega)}\right)} \\ &= e^{j\arctan\left(\frac{(1-a^2)\sin(\omega)}{(1+a^2)\cos(\omega)-2a}\right)}, \end{aligned} \tag{5.7}$$

É preciso recorrer à equação 5.8 para o último passo.

$$\tan(a-b) = \frac{\tan(a)-\tan(b)}{1-\tan(a)\tan(b)}. \tag{5.8}$$

A equação 5.7 mostra que o comportamento em fase de um filtro passa-tudo é não-linear. Em alternativa, da seguinte equação conclui-se também que a fase do passa-tudo é não-linear

$$H(e^{j\omega}) = \frac{z^{-1}-a}{1-az^{-1}}\Big|_{z=e^{j\omega}} = \frac{z-a}{z(1-az^{-1})}\Big|_{z=e^{j\omega}} = e^{-j\omega-j2\arctan\left(\frac{a\sin(\omega)}{1-a\cos(\omega)}\right)}. \tag{5.9}$$

Uma aplicação característica de aplicação de um sistema passa-tudo (5.5) é a transformação bilinear

$$\Psi^{-1} = \frac{z^{-1}-a}{1-az^{-1}}. \tag{5.10}$$

A transformação projecta o plano z no plano Ψ, sendo $z = e^{j\omega}$ e $\Psi = e^{j\theta}$ obtém-se

$$\theta = \omega + 2\arctan\left(\frac{a\sin(\omega)}{1-a\cos(\omega)}\right), \tag{5.11}$$

resultando numa projecção não-linear de frequências ω em frequências θ. A transformada de Fourier em frequência distorcida (Warped Discrete Fourier Transform (WDFT)) define-se como

$$X(\Psi) = \sum_{n=0}^{N-1} x(n)\Psi^{-n} = \sum_{n=0}^{N-1} x(n)[H(z)]^n, \tag{5.12}$$

e é obtida substituindo os operadores de atraso z^{-1} por filtros passa-tudo $H(z)$, resultando numa amostragem não-uniforme do plano z, concentrando mais os pontos junto de $\omega = 0$ para $a > 0$, e concentrando-os junto de $\omega = \pi$ para $a < 0$.

Um banco de N filtros pode assim ser utilizado para realizar o mapeamento não-linear. A figura 5.5 ilustra este conceito, o sistema é composto por um banco de filtros para análise e síntese, os filtros de análise são obtidos pela modulação em frequência do filtro protótipo passa-baixo com coeficientes $p(n), \quad 0 \leq n \leq N-1$, e os de síntese são obtidos pela modulação do filtro protótipo passa-baixo com coeficientes $f(n), \quad 0 \leq n \leq N-1$. A transformação para o domínio das frequências é realizada com uma ODFT [1], enquanto que para voltar ao domínio dos tempos é realizada a sua inversa IODFT. Os processos de análise e síntese são realizados com 50% de sobreposição entre tramas adjacentes devido ao facto de o factor de decimação e interpolação ser

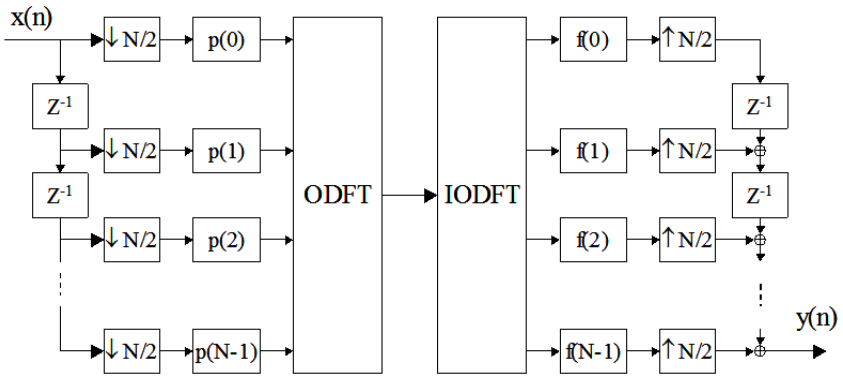

Figura 5.5: Estrutura do banco de N filtros (adaptada de Ferreira (2007))

N/2. Se os filtros respeitarem as condições de reconstrução perfeita e não existir modificação espectral, então a saída é igual a entrada a menos de um atraso. Para que esta estrutura realize uma análise não uniforme é necessário realizar a mudança dos operadores de atraso z^{-1} por Ψ^{-1}, ver figura 5.6. Uma vez que não é exercida qualquer modificação espectral entre a ODFT e a IODFT, estas podem ser suprimidos sem que exista qualquer alteração do sistema, resultando na estrutura da figura 5.7. Na síntese não se efectua a mudança dos atrasos para evitar a correcção da projecção

[1] A utilidade da ODFT (em vez da DFT) decorre de exibir simetria em N/2 pontos (em vez de N/2+1 pontos) e de facilitar a análise espectral quando conjugada com a janela seno Ferreira (2001)

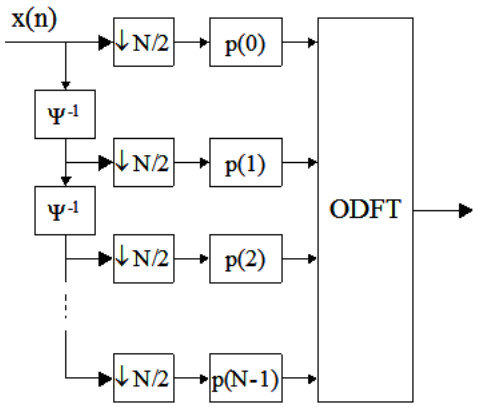

Figura 5.6: Estrutura do banco de N filtros para realizar *frequency warping* (adaptada de Ferreira (2007))

não-linear. Uma vez que

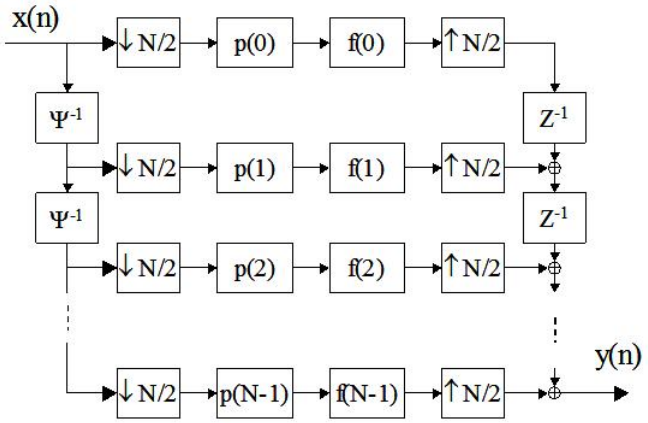

Figura 5.7: Estrutura do banco de N filtros sem as operações de transformação ODFT e IODFT (adaptada de Ferreira (2007))

$$p(n) = f(n) = \sin\left[\left(\frac{pi}{N}\right)(n+0,5)\right], \quad 0 \leq n \leq N-1, \tag{5.13}$$

as operações intermédias entre os blocos decimadores e interpoladores resume-se a uma multiplicação por cada ramo. Sendo assim, o funcionamento do sistema resume-se a

i) fazer com que o sinal percorra toda a cascata de filtros passa-tudo,

ii) de N/2 em N/2 amostras recolher o sinal da cadeia e realizar uma multiplicação por um vector de comprimento N,

iii) efectuar um *overlap add* com 50% de sobreposição do resultado anterior para se obter o sinal de saída.

Uma vez que

$$H(z) = \frac{z^{-1} - a}{1 - az^{-1}}, \tag{5.14}$$

a sua equação as diferenças é

$$y(n) = x(n-1) - ax(n) + ay(n-1). \tag{5.15}$$

A representação da sua estrutura directa e canónica são ilustradas na figura 5.8, por sua vez as equações às diferenças (5.16) representam a estrutura canónica.

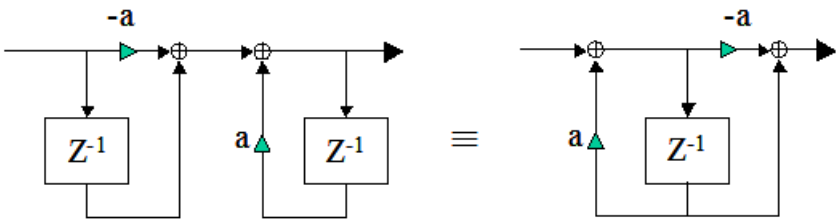

Figura 5.8: Estrutura directa e canónica do sistema passa-tudo (adaptada de Ferreira (2007))

$$\begin{aligned} w(n) &= x(n) + aw(n-1) \\ y(n) &= w(n-1) - aw(n) \end{aligned} \tag{5.16}$$

Após a exposição teórica de Ferreira (2007) e a descrição do seu sistema, estamos em condições de passar para o nosso algoritmo, o qual é ilustrado na figura 5.9 através um diagrama de blocos. Da análise da figura podemos constatar que a sua estrutura é idêntica ao primeiro algoritmo descrito em 5.1.1 para o canal 1. O canal 2 por sua vez apresenta uma estrutura ligeiramente diferente, neste captamos o sinal entregue pela análise e realizamos a FFT como no algoritmo anterior. Com o sinal no domínio das frequências obtém-se a potência das duas regiões (5.1), e verifica-se se a relação de potências satisfaz a condição (5.2), ou seja, se é maior que 1.42. No caso de ser maior aplica-se o sinal que foi captado da análise e aplica-se ao sistema de *frequency warping*, que por sua vez irá aplicar as equações (5.16) para realizar a compressão do espectro e a translação para a região de 1.5 kHz a 2 kHz.O sinal depois de alterado é reconstruído com o *overlap add* de 50% e uma filtragem passa-banda para seleccionar a região de 1.5 kHz a 2 kHz. O algoritmo termina com a soma dos sinais dos dois canais.

5.1.3 Translação ou Compressão

O algoritmo de translação ou compressão surgiu depois de se constatar que há zonas do espectro nas consoantes que apresentam um pico de energia relevante na zona acima dos 2 kHz.

O canal 1 do algoritmo é igual aos anteriores, o canal 2 também o é com a excepção do bloco que realiza a translação ou compressão (ver figura 5.10), este bloco difere do algoritmo de compressão e translação no sentido em que, em primeiro lugar procura o pico com mais energia na região acima dos 2 kHz e os máximos locais também. Sabendo qual é o máximo verifica-se se existem mais picos que se possam revelar importantes, e para isso são comparados os máximos locais com 75% do máximo. Se existir mais algum pico então é realizada compressão e translação

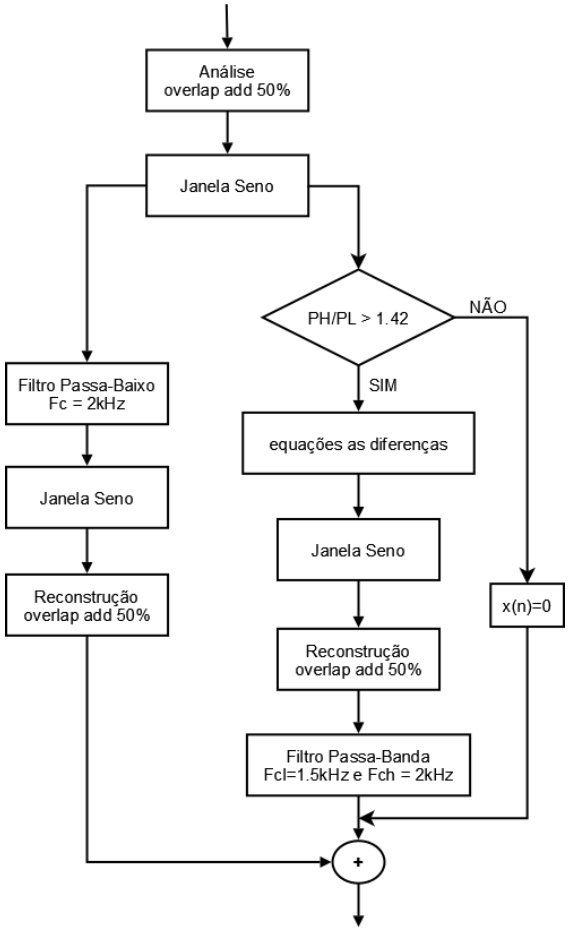

Figura 5.9: Diagrama de blocos do algoritmo de *frequency warping*.

do primeiro algoritmo, se não é realizada uma translação do pico máximo e mais seis bins de cada lado, mantendo sempre a paridade dos bins transladados (ver figura 5.11). A partir de aqui o sinal é tratado de forma igual ao primeiro algoritmo.

5.1.4 Compressão e Translação ou Síntese de Harmónicas

Este algoritmo aparece devido ao facto de as vogais /i/ e /e/ terem formantes relevantes acima da região dos 2 kHz, devido a isso, decidiu-se adicionar informação na região de 1.5 kHz a 2 kHz com características tais como, a amplitude e a fase.

A estrutura do algoritmo segue a lógica dos anteriores (ver figura 5.12), ou seja, uma divisão em dois canais. O sinal é analisado em tramas de 256 elementos e sobreposição de 50% e multiplicadas pela janela seno, visto que agora os dois canais precisam do sinal no domínio das frequências, então é realizada uma FFT de 256 pontos, sendo o resultado entregue aos dois canais. o canal 1 realiza uma análise ao espectro para poder determinar se o que consta na trama pode ser uma vogal /i/ ou uma vogal /e/. As duas vogais em questão apresentam um vale entre sensivelmente 1 kHz e 2 kHz no seu espectro, assim sendo calculam-se as seguintes potências:

1. entre 0 e 1 kHz passando a ser designada por PLF,

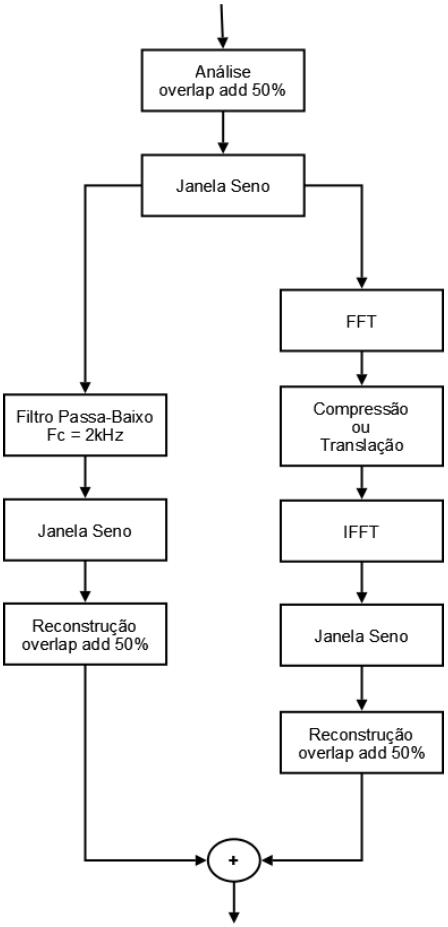

Figura 5.10: Diagrama de blocos do algoritmo de translação ou compressão.

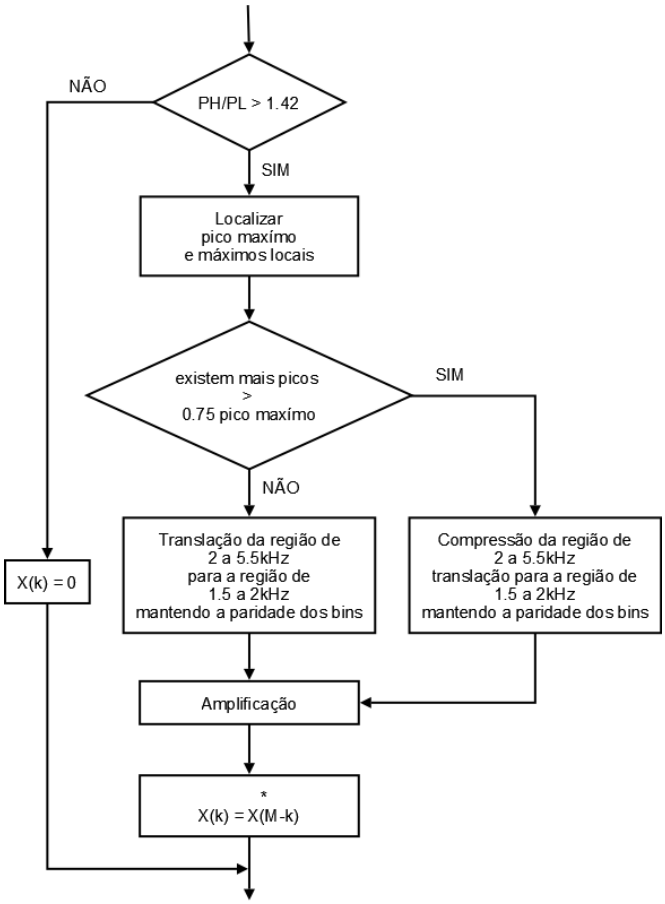

Figura 5.11: Diagrama de blocos da translação ou compressão.

2. entre 1 kHz e 2 kHz passando a ser designada por PMF,

3. entre 2 kHz e 4 kHz passando a ser designada por PHF.

A seguinte equação relaciona as potências de maneira que seja possível encontrar uma vogal /i/

$$\frac{PHF}{PMF} > 3.1 \qquad \frac{PLF}{PMF} > 2.1, \tag{5.17}$$

por sua vez a as seguintes relações permitem que seja possível encontrar uma vogal /e/

$$\frac{PHF}{PMF} > 2 \qquad \frac{PLF}{PMF} > 1. \tag{5.18}$$

Se uma das relações, a (5.17) ou (5.18) for satisfeita então são sintetizadas três sinusóides, uma com a amplitude e fase do pico de energia mais elevado da zona acima dos 2 kHz e as outras duas com os outros dois máximos, o da esquerda e o da direita do máximo. Estas sinusóides são criadas na zona de 1.5 kHz a 2 kHz, esta zona só nos permite utilizar 12 bins o que nos deixa margem de 4 quatro bins entre as sinusóides para que não interferiram entre elas. Só falta recriar a parte

das frequências negativas através de (5.4) e realizar a IFFT para voltar para o domínio do tempo e multiplicar pela janela seno. A reconstrução do sinal é feita com *overlap add* com sobreposição de 50% entre tramas. O canal 2 por sua vez é exactamente igual ao algoritmo compressão/translação, ou seja, o sinal é processado de igual modo. O sinal de saída é mais uma vez a soma dos dois canais. A figura 5.12 ilustra o algoritmo implementado, por outro lado a figura 5.13 apresenta o

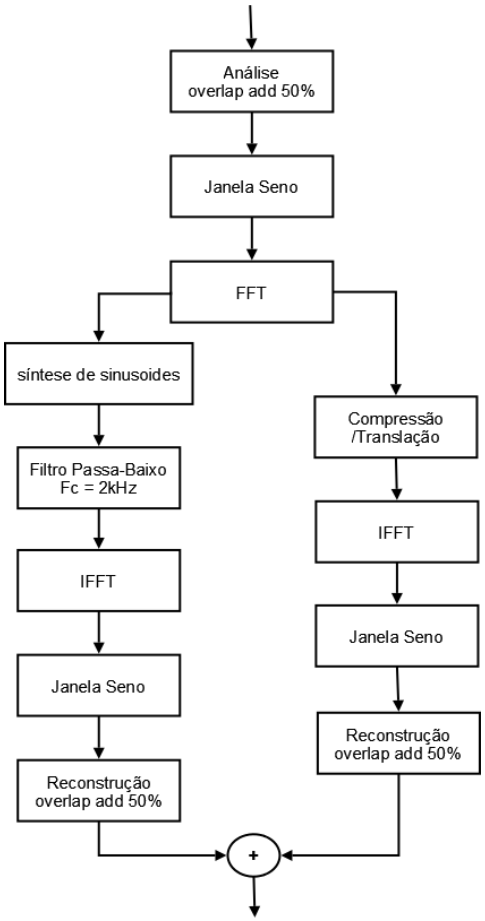

Figura 5.12: Diagrama de blocos do algoritmo de compressão/translação e síntese de harmónicas.

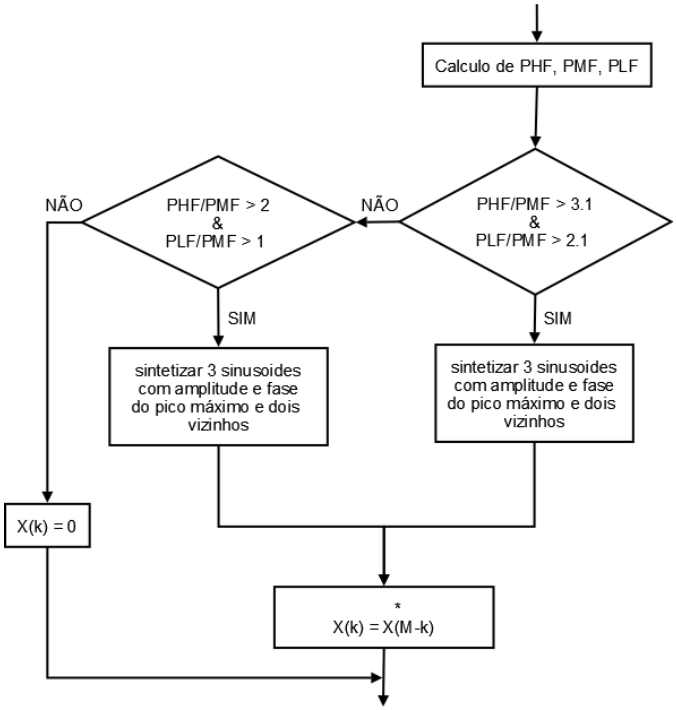

Figura 5.13: Diagrama de blocos da síntese de harmónicas.

diagrama de blocos da síntese do diagrama de blocos da figura 5.12

5.2 Simulação

Nesta secção são apresentados os resultados obtidos das simulações dos algoritmos implementados, para efectuar as simulações foram utilizados ficheiro de áudio que contêm palavras em Português Europeu, e uma frequência de amostragem de 11.025 kHz.

5.2.1 Compressão e Translação

A simulação do algoritmo de compressão/translação foi realizado com três palavras, sendo elas “cores”, “preferidos”, e “são”, a figura 5.14 representa três espectrogramas, o primeiro é um

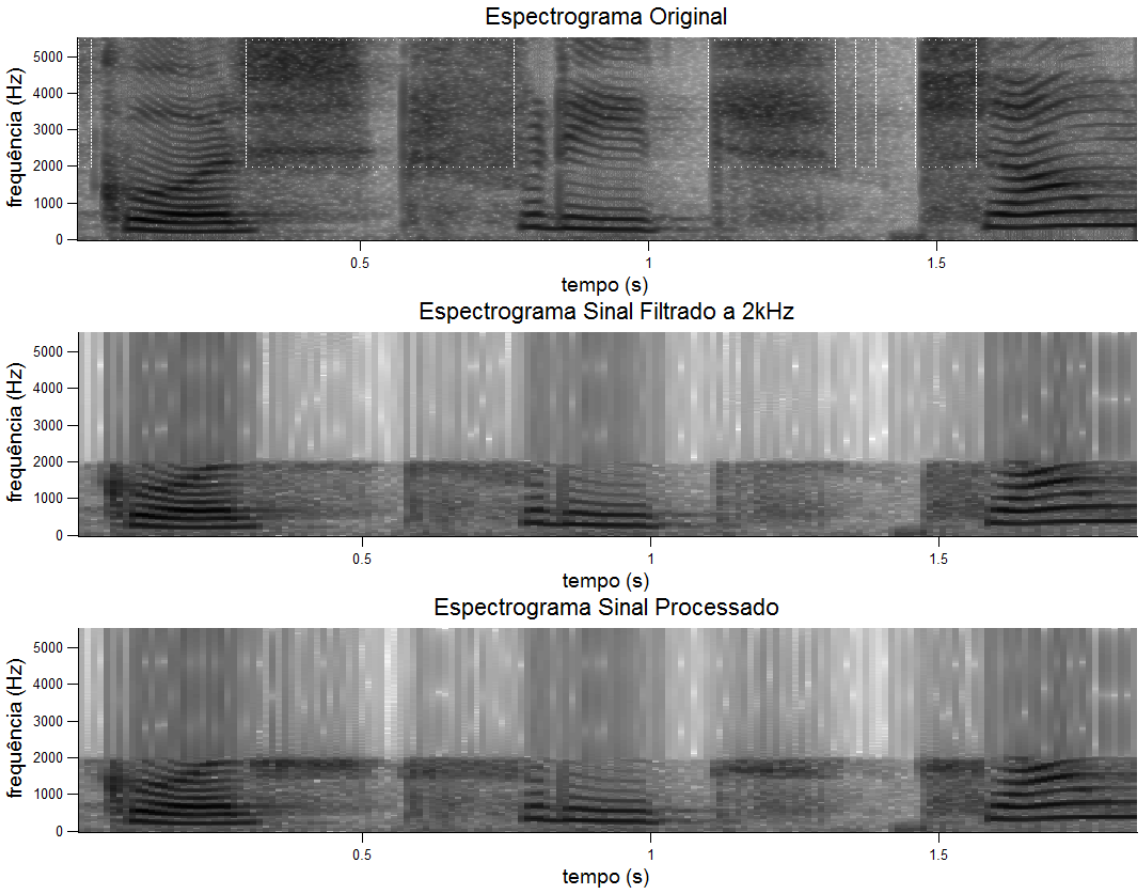

Figura 5.14: A figura superior representa um espectrograma do sinal original, a do meio representa o sinal filtrado, e a ultima o sinal depois de processado.

espectrograma do ficheiro original que contém as três palavras, neste espectrograma encontram-se uns rectângulos brancos que representam as zonas que devem ser comprimidas e transladas. O espectrograma do meio apresenta o sinal filtrado a 2 kHz, com esta filtragem pretende-se simular a perda auditiva acima de 2 kHz. Finalmente o último espectrograma reproduz o sinal processado, comparando o espectrograma da filtragem com este podemos observar no alinhamento dos rectângulos uma maior concentração de energia na região de 1.5 kHz a 2 kHz.

5.2.2 Frequency Warping e Translação

O algoritmo *frequency warping* foi simulado com o mesmo ficheiro que em 5.2.1, a figura 5.15 apresenta a mesma estrutura que na simulação compressão e translação (5.2.1). Da mesma forma

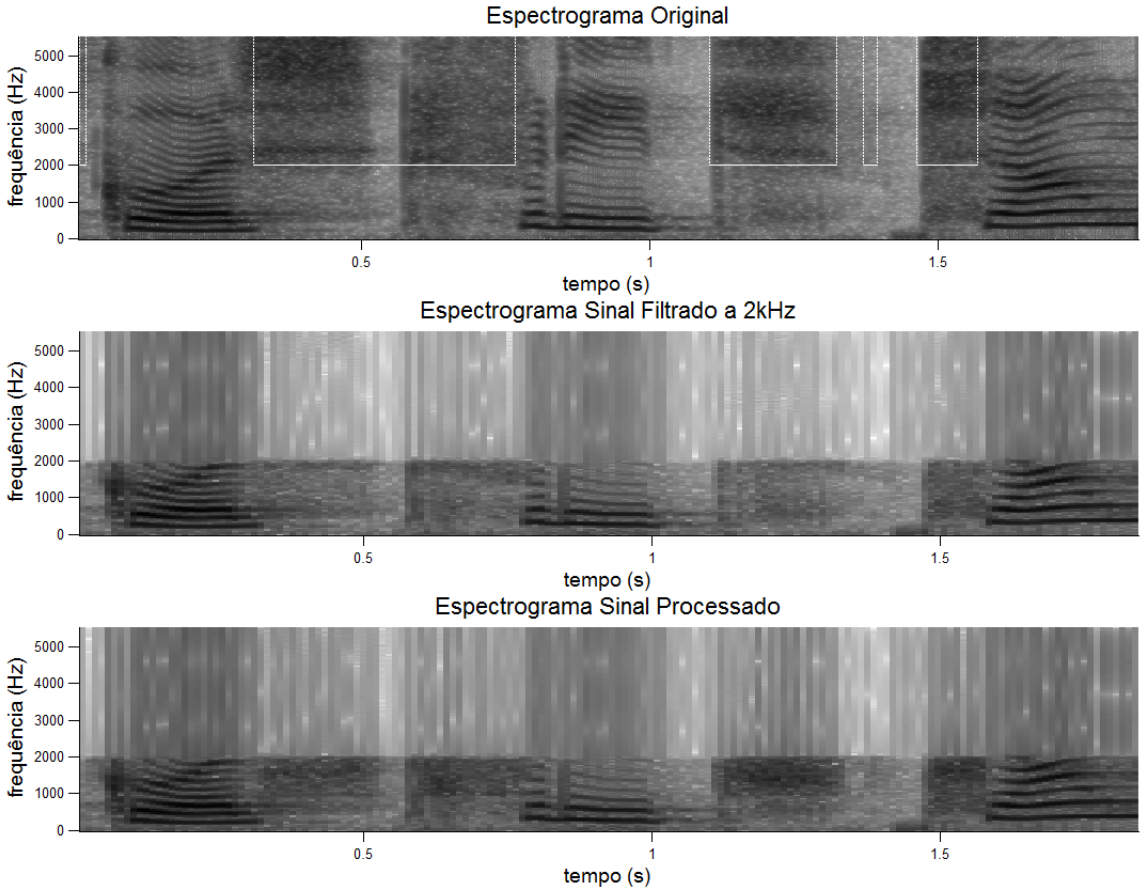

Figura 5.15: A figura superior representa um espectrograma do sinal original, a do meio representa o sinal filtrado, e a ultima o sinal depois de processado.

se se comparar o espectrograma da filtragem com o do sinal processado, verifica-se que no alinhamento dos rectângulos o espectrograma do sinal processado apresenta um acréscimo de energia na região de 1.5 kHz a 2 kHz.

5.2.3 Translação ou Compressão

A simulação deste algoritmo também foi realizada com recurso ao ficheiro que contém as palavras utilizadas nos outros dois algoritmos (compressão e translação 5.2.1 e *frequency warping* 5.2.2), à figura 5.16 aplica-se o que foi referido acima.

5.2.4 Translação, Compressão e Síntese de Harmónicas

Este algoritmo foi testado com um ficheiro que contém as palavras “tempo”, “real”, “vida”, a figura 5.17 ilustra os espectrogramas com os resultados obtidos. O espectrograma superior apresenta o sinal original no qual é marcado com um rectângulo a vogal /e/, por sua vez a vogal /i/ não foi detectada. O resultado da síntese pode ser observado no espectrograma inferior, verificando-se

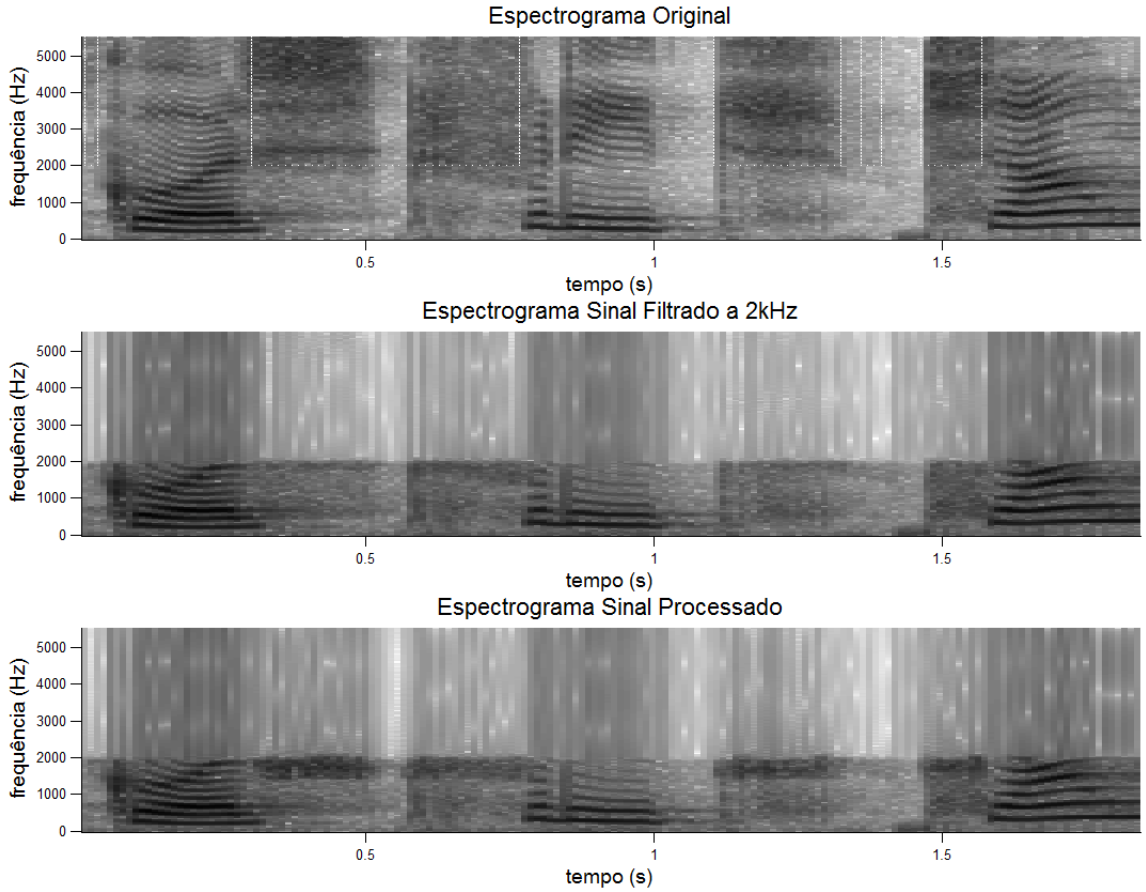

Figura 5.16: A figura superior representa um espectrograma do sinal original, a do meio representa o sinal filtrado, e a ultima o sinal depois de processado.

na zona de 1.5 kHz a 2 kHz alinhada com os rectângulos que surgiu um aumento de energia, mas também surgiu uma alteração harmónica.

5.3 Discussão

No seguimento das simulações realizadas com os algoritmos de processamento das altas frequências decorreu à análise dos resultados obtidos. O algoritmo compressão e translação cumpriu o primeiro objectivo que era de preencher a zona definida (1.5 kHz a 2 kHz) com o espectro das altas frequências. O segundo objectivo era contribuir para que a discriminação das consoantes melhorasse. Os testes efectuados com várias palavras revelaram que este algoritmo não foi totalmente eficaz com todas as palavras de teste, e introduziu uma ligeira distorção quando se ouve a parte que foi processada. Contudo na consoante /s/ identifica-se um ligeiro reforço.

Os resultados do algoritmo de *frequency warping* foram semelhantes aos apresentados pelo algoritmo de compressão e translação, foi possível mapear a zona de altas frequências para a nossa zona de destino nas baixas frequências. Ao testar o algoritmo com diferentes palavras, o

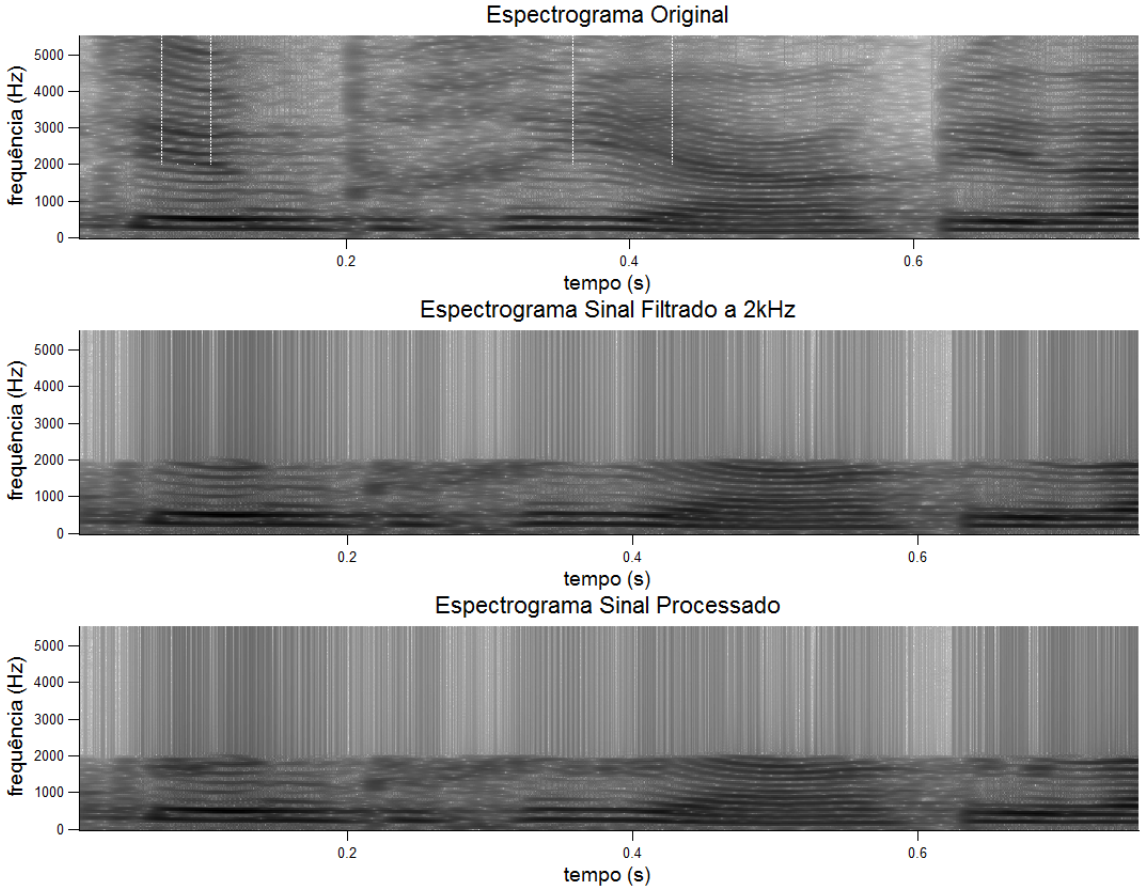

Figura 5.17: A figura superior representa um espectrograma do sinal original, a do meio representa o sinal filtrado, e a ultima o sinal depois de processado.

som produzido é semelhante ao produzido pelo algoritmo anterior mas com ligeiro aumento na distorção.

Por sua vez os resultados do algoritmo de translação ou compressão não apresentaram qualquer melhoramento em relação aos dois anteriores, confirma-se contudo que é capaz transladar informação das altas frequências para a região estipulada. A sua sonoridade é muito semelhante à produzida pelo algoritmo de compressão e translação.

A análise do algoritmo de compressão translação e síntese de harmónicas revelou que o algoritmo não é totalmente robusto na detecção das vogais /i/ e /e/, visto que na simulação não classificou uma vogal /i/. No entanto quando detectou a vogal /e/ realizou a síntese das sinusóides na zona 1.5 kHz a 2 kHz. O resultado prático desta tentativa de criar reforço destas vogais não produziu o resultado esperado, visto que para além de não produzir melhoramento ainda criou mais desconforto ao ouvido.

Importa referir que para o teste auditivo, realizado com um ouvinte, em primeiro lugar era ouvido o som filtrado a 2 kHz por um filtro passa-baixo (como se pode observar nas figuras anteriores), com 80 dB de atenuação na banda de corte, e em segundo lugar era ouvido o som resultante do processamento.

5.4 Conclusão

Analisando os resultados obtidos dos algoritmos é possível concluir que alguns deles apresentaram resultados parcialmente satisfatórios, ou seja, não tiveram uma resposta totalmente eficaz a todas as palavra que lhes sejam aplicadas. Contudo o algoritmo de compressão e translação foi o que apresentou um desempenho mais promissor, revelando resultados satisfatórios. Embora eles não resolvam todas as situações com que são confrontados, eles também não prejudicaram a inteligibilidade dos sons, com excepção para o algoritmo de síntese que parece criar um pequeno desconforto ao ouvido nas vogais /e/ e /i/. Este desconforto advém do facto de ser alterada a estrutura harmónica do som quando se adiciona a informação criada pelas sinusóides.

Capítulo 6

Conclusões e Trabalho Futuro

6.1 Resultados e Contribuições

O trabalho aqui descrito teve por base o trabalho realizado por Fraga et al. (2008), visto que tinha apresentado resultados satisfatórios na discriminação de consoantes do Português do Brasil. Neste trabalho realizamos quatro algoritmos. Um algoritmo de compressão e translação, *frequency warping*, compressão ou translação e o último que é o de compressão e translação com síntese de harmónicas. Um aspecto que se pensou que poderia aportar algum melhoramento seria o facto de se combinar a translação quando só existe um pico relevante de energia acima dos 2 kHz ou realizar uma compressão quando existe mais que um, os resultados obtidos não foram relevantes pois não apresentou melhoramentos satisfatórios. Outra proposta que se imaginava que teria alguma importância, foi sintetizar sinusóides a partir da estrutura harmónica para as vogais /i/ e /e/, uma vez que estas apresentam formantes acima dos 2 kHz. Os resultados práticos revelaram que não pode ser realizada desta maneira porque produziu desconforto ao ouvido.

6.2 Trabalho Futuro

A ideia de reforçar as vogais /i/ e /e/ parece pertinente, visto que apresentam formantes acima dos 2 kHz, um melhoramento possível será procurar ou melhorar a síntese de sinusóides, fazendo com que elas sigam a estrutura harmónica na região de 1.5 kHz a 2 kHz. Outro aspecto a melhorar será tornar o sistema de detecção destas vogais mais robusto.

Uma implementação em tempo real dos algoritmos implementados numa plataforma com DSP (Processador Digital de Sinal) para verificar a viabilidade dos algoritmos.

Anexo A

Código Compressão e Translação

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Algorithm to make the frequency compression %
%                                              %
% Author : Ricardo Costa                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all;
clear all;
clc;

Fpass = 1900;  %LP Filter
Fpass1 = 1450; %BP Filter
Fpass2 = 1950; %BP Filter

Apass = 0.1;
Astop = 80;

Nfft = 256;
L = Nfft;

% Window Sine
win = sin((pi/Nfft)*((0:Nfft-1)+0.5));
win = win';

% Read audio signal from wave file
[in, Fs, NBITS] = wavread('teste');

in(129:length(in)+128) = in(1:length(in));
in(1:128) = zeros(128,1);

if mod(length(in),Nfft) ~= 0
    while mod(length(in),Nfft) ~= 0
        in(length(in)+1) = 0;
    end
end
```

```
% Filter coeficients
[Blp, Alp] = lowPassCoefs(Fs, Fpass, Apass, Astop);

% Initialize variables
channel1 = zeros(length(in),1);
channel2 = zeros(length(in),1);
out      = zeros(length(in),1);
X        = zeros(Nfft,1);
newX     = zeros(Nfft,1);
maxl     = zeros(Nfft,1);
step = Fs/Nfft;
initf = round(1500/step);
endf = round(2000/step);
bw = round((endf - initf)/2);
dataplot = [zeros(Nfft,1) zeros(Nfft,1) zeros(Nfft,1)];
frames = length(in)/L;

% k=0
% Channel 1
% High frequencies filtered
buf = lowPassFilter(in(1:L).*win, Blp, Alp);
channel1(1:L) = buf.*win;
% Channel 2
% Frequency domain
X = fft((in(1:L).*win), Nfft);
% Calculate power
lowpower = sum(abs(X(1:endf)).^2)/endf;
highpower = sum(abs(X(endf:(Nfft/2)+1)).^2)/((Nfft/2)+1-endf);

iter = 1;
if abs(highpower/lowpower) > 1.42
    flagi = 0;
    for bin=round(2000/step):((Nfft)/2)+1
        fout = round(0.143*(bin*43)+1213.5);
        bout = round(fout/43);
        if mod(fout,43) == 0
            if mod(bin,2) == 0
                if mod(bout,2) == 0
                    if flagi == 0
                        flagi = 1;
                        dataplot(iter,:) = [bin (1/Fs)*(L/2) (1/Fs)*(L/2)+(1/Fs)*(L/2)];
                        iter = iter + 1;
                    else
                        dataplot(iter,:) = [bin (1/Fs)*(L/2) (1/Fs)*(L/2)+(1/Fs)*(L/2)];
                    end
                    newX(bout) = X(bin)*1.2;
                else
                    if flagi == 0
                        flagi = 1;
```

```
                    dataplot ( iter ,:) = [bin (1/Fs)*(L/2) (1/Fs)*(L/2)+(1/Fs)*(L/2)];
                    iter = iter + 1;
                else
                    dataplot ( iter ,:) = [bin (1/Fs)*(L/2) (1/Fs)*(L/2)+(1/Fs)*(L/2)];
                end
                newX(bout) = X(bin-1)*1.2;
            end
        else
            if mod(bout,2) == 0
                if flagi == 0
                    flagi = 1;
                    dataplot ( iter ,:) = [bin (1/Fs)*(L/2) (1/Fs)*(L/2)+(1/Fs)*(L/2)];
                    iter = iter + 1;
                else
                    dataplot ( iter ,:) = [bin (1/Fs)*(L/2) (1/Fs)*(L/2)+(1/Fs)*(L/2)];
                end
                newX(bout) = X(bin-1)*1.2;
            else
                if flagi == 0
                    flagi = 1;
                    dataplot ( iter ,:) = [bin (1/Fs)*(L/2) (1/Fs)*(L/2)+(1/Fs)*(L/2)];
                    iter = iter + 1;
                else
                    dataplot ( iter ,:) = [bin (1/Fs)*(L/2) (1/Fs)*(L/2)+(1/Fs)*(L/2)];
                end
                newX(bout) = X(bin)*1.2;
            end
        end
    end
else
    newX = zeros(Nfft ,1);
end
% Get negative frequencies
newX(Nfft:-1:(Nfft/2)+2) = conj(newX(2:(Nfft /2)));
% Return to time domain
X = real( ifft (newX,Nfft));
newX = zeros(Nfft ,1);
channel2(1:L) = X(1:L).*win;

% k>0
for k=1:(2*frames)-2
    % Channel1
    % High frequencies filtered
    buf = lowPassFilter (in(k*(L/2)+1:k*(L/2)+L).*win, Blp, Alp);
    % Overlap add 50%
    channel1(k*(L/2)+1:k*(L/2)+L) = (buf.*win) + channel1(k*(L/2)+1:k*(L/2)+L);
    % Channel2
    % Frequency domain
```

```
X = fft ((in(k*(L/2)+1:k*(L/2)+L).*win), Nfft );
% Calculate power
lowpower = sum(abs(X(1:endf )).^2)/ endf;
highpower = sum(abs(X(endf:(Nfft /2)+1)).^2)/(( Nfft/2)+1-endf);

if abs(highpower/lowpower) > 1.42
    flagi = 0;
    for bin=round(2000/step ):(( Nfft )/2)+1
        fout = round(0.143*(bin*43)+1213.5);
        bout = round(fout/43);
        if fout-bout*43 <= 2
            if mod(bin,2) == 0
                if mod(bout,2) == 0
                    if flagi == 0
                        flagi = 1;
                        dataplot ( iter ,:) = [bin k*(1/Fs)*(L/2) k*(1/Fs)*(L/2)+(1/Fs)*(L/2)];
                        iter = iter + 1;
                    else
                        dataplot ( iter ,:) = [bin k*(1/Fs)*(L/2) k*(1/Fs)*(L/2)+(1/Fs)*(L/2)];
                    end
                    newX(bout) = X(bin)*1.2;
                else
                    if flagi == 0
                        flagi = 1;
                        dataplot ( iter ,:) = [bin k*(1/Fs)*(L/2) k*(1/Fs)*(L/2)+(1/Fs)*(L/2)];
                        iter = iter + 1;
                    else
                        dataplot ( iter ,:) = [bin k*(1/Fs)*(L/2) k*(1/Fs)*(L/2)+(1/Fs)*(L/2)];
                    end
                    newX(bout) = X(bin-1)*1.2;
                end
            else
                if mod(bout,2) == 0
                    if flagi == 0
                        flagi = 1;
                        dataplot ( iter ,:) = [bin k*(1/Fs)*(L/2) k*(1/Fs)*(L/2)+(1/Fs)*(L/2)];
                        iter = iter + 1;
                    else
                        dataplot ( iter ,:) = [bin k*(1/Fs)*(L/2) k*(1/Fs)*(L/2)+(1/Fs)*(L/2)];
                    end
                    newX(bout) = X(bin-1)*1.2;
                else
                    if flagi == 0
                        flagi = 1;
                        dataplot ( iter ,:) = [bin k*(1/Fs)*(L/2) k*(1/Fs)*(L/2)+(1/Fs)*(L/2)];
                        iter = iter + 1;
                    else
                        dataplot ( iter ,:) = [bin k*(1/Fs)*(L/2) k*(1/Fs)*(L/2)+(1/Fs)*(L/2)];
                    end
```

```
                    newX(bout) = X(bin)*1.2;
                end
            end
        end
    end
    iter = iter + 1;
else
    newX = zeros(Nfft,1);
end
% Get negative frequencies
newX(Nfft:-1:(Nfft/2)+2) = conj(newX(2:(Nfft/2)));
% Return to time domain
X = real( ifft (newX,Nfft));
newX = zeros(Nfft,1);
% Overlap add 50%
channel2(k*(L/2)+1:k*(L/2)+L) = (X(1:L).*win) + channel2(k*(L/2)+1:k*(L/2)+L);
end
% Filtered signal
wavwrite(channel1/max(abs(channel1)),Fs,NBITS,'channel1');
% Output file
out = channel1 + channel2;
wavwrite(out/max(abs(out)),Fs,NBITS,'compression');

flaginit = 0;
flagend = 1;
ptr = 1;
% Find rectangles boundaries
for jj=1:length( dataplot )-1
    if dataplot ( jj ,:)==0
        break;
    end
    if flaginit == 0 && flagend == 1
        boundaries( ptr ) = jj ;
        ptr = ptr + 1;
        flaginit = 1;
        flagend = 0;
    end
    if flagend == 0 && flaginit == 1
        if abs( dataplot ( jj+1,2)-dataplot ( jj ,3)) > (1/Fs)*Nfft
            boundaries( ptr ) = jj ;
            ptr = ptr + 1;
            flaginit = 0;
            flagend = 1;
        end
    end
end

subplot (3,1,1);
% Plot original signal
```

```
[B,f,t] = spectrogram(in,256,250,Nfft,Fs);
imagesc(t,f,20*log10(abs(B)));
 title ('Espectrograma␣Original','FontSize',16);
xlabel('tempo␣(s)','FontSize',14);
ylabel('frequência␣(Hz)','FontSize',14);
set(gca, 'Box', 'off');
axis xy
set(gca, 'TickDir', 'out', 'XTick',0:0.5:length(in)*(1/Fs)-(1/Fs), 'YTick', 0:1000:5500);
lgrays=zeros(100,3);
for i=1:100
    lgrays(i,:) = 1-i/100;
end
colormap(lgrays);
hold on;
% Plot white rectangles
for kk=1:2:length(boundaries)-1
    max = 0;
    min = 100000;
    for ll=boundaries(kk):boundaries(kk+1)
        if (dataplot(ll,1)+ bw)*(Fs/Nfft) > max
            max = (dataplot(ll,1))*(Fs/Nfft);
        end
        if (dataplot(ll,1)- bw)*(Fs/Nfft) < min
            min = (dataplot(ll,1))*(Fs/Nfft);
        end
    end
    plot(dataplot(boundaries(kk),2):(1/Fs)*(L/2):dataplot(boundaries(kk+1),3),min,'w')
    hold on;
    plot(dataplot(boundaries(kk),2):(1/Fs)*(L/2):dataplot(boundaries(kk+1),3),max,'w')
    hold on;
    plot(dataplot(boundaries(kk),2),min:Fs/Nfft:max,'w')
    hold on;
    plot(dataplot(boundaries(kk+1),3),min:Fs/Nfft:max,'w')
    hold on;
end
hold off;
% Plot filtered signal
subplot(3,1,2);
[B,f,t] = spectrogram(channel1,250,128,Nfft,Fs);
imagesc(t,f,20*log10(abs(B)));
 title ('Espectrograma␣Sinal␣Filtrado␣a␣2kHz','FontSize',16);
xlabel('tempo␣(s)','FontSize',14);
ylabel('frequência␣(Hz)','FontSize',14);
set(gca, 'Box', 'off');
axis xy
set(gca, 'TickDir', 'out', 'XTick',0:0.5:length(in)*(1/Fs)-(1/Fs), 'YTick', 0:1000:5500);
lgrays=zeros(100,3);
for i=1:100
    lgrays(i,:) = 1-i/100;
```

```
end
colormap(lgrays);
% Plot compressed signal
subplot (3,1,3);
[B,f,t] = spectrogram(out,250,128,Nfft,Fs);
imagesc(t,f,20*log10(abs(B)));
 title ('Espectrograma Sinal Processado','FontSize',16);
xlabel('tempo (s)','FontSize',14);
ylabel('frequência (Hz)','FontSize',14);
set(gca, 'Box', 'off');
axis xy
set(gca, 'TickDir', 'out', 'XTick',0:0.5:length(in)*(1/Fs)-(1/Fs), 'YTick', 0:1000:5500);
lgrays=zeros(100,3);
for i=1:100
    lgrays(i,:) = 1-i/100;
end
colormap(lgrays);
```

Anexo B

Código Frequency Warping

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Algorithm to make the frequency warping/ shift %
%                                                %
% Author : Ricardo Costa                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all;
clear all;
clc;

Fpass = 1900;  %LP Filter
Fpass1 = 950; %BP Filter
Fpass2 = 1950; %BP Filter

Apass = 0.1;
Astop = 80;

Nfft = 256;
L = Nfft;

%Window created with sin(x)
win = sin((pi/Nfft)*((0:Nfft-1)+0.5));

%Read audio signal from wave file

[in, Fs, NBITS] = wavread('C:\Users\Ricardo Costa\Documents\DISSERTACAO\matlab\audio\teste');

in(129:length(in)+128) = in(1:length(in));
in(1:128) = zeros(128,1);

if mod(length(in), Nfft) ~= 0
    while mod(length(in), Nfft) ~= 0
        in(length(in)+1) = 0;
    end
end
```

```
%Filter coeficients
[Blp, Alp] = lowPassCoefs(Fs, Fpass, Apass, Astop);
[Bbp, Abp] = bandPassCoefs(Fs, Fpass1, Fpass2, Apass, Astop);

% Initialize variables
channel1 = zeros(length(in),1);
channel2 = zeros(length(in),1);
out      = zeros(length(in),1);
X        = zeros(Nfft,1);
newX     = zeros(Nfft,1);
maxl     = zeros(Nfft,1);
step = Fs/Nfft;
initf = round(1500/step);
endf = round(2000/step);
bw = round((endf - initf)/2);
dataplot = [zeros(Nfft,1) zeros(Nfft,1) zeros(Nfft,1)];
iter = 1;
frames = length(in)/L;
alfa = -0.55;
wmem = zeros(1,L);
idata = zeros(1,L);

% k=0
% Channel 1
% High frequencies filtered
buf = lowPassFilter(in(1:L).*win, Blp, Alp);
channel1(1:L) = buf.*win;
% Channel 2
% Frequency domain
X = fft((in(1:L).*win'), Nfft);
% Calculate power
lowpower = sum(abs(X(1:endf)).^2)/endf;
highpower = sum(abs(X(endf:(Nfft/2)+1)).^2)/((Nfft/2)+1-endf);

if abs(highpower/lowpower) > 1.42
    buffer = in(1:L);
    for li=1:L/2
        idata(1,1) = buffer(li,1);
        flagi = 0;
        if flagi == 0
            flagi = 1;
            dataplot(iter,:) = [0 (1/Fs)*(L/2) (1/Fs)*(L/2)+(1/Fs)*(L/2)];
            iter = iter + 1;
        else
            dataplot(iter,:) = [0 (1/Fs)*(L/2) (1/Fs)*(L/2)+(1/Fs)*(L/2)];
        end
        % Frequency Warping
        for lw=2:L
            dtmp = idata(1,lw-1) + alfa*wmem(lw);
```

```
            idata (1,lw) = wmem(lw) - alfa*dtmp;
            wmem(lw) = dtmp;
        end
    end
    buffer=idata (1,L:-1:1).*win;
    channel2(1:L) = buffer';
end

% k>0
for k=1:(2*frames)-2
    % Channel1
    % High frequencies filtered
    buf = lowPassFilter (in(k*(L/2)+1:k*(L/2)+L).*win, Blp, Alp);
    % Overlap add 50%
    channel1(k*(L/2)+1:k*(L/2)+L) = (buf.*win) + channel1(k*(L/2)+1:k*(L/2)+L);
    % Channel2
    % Frequency domain
    X = fft ((in(k*(L/2)+1:k*(L/2)+L).*win'), Nfft);
    % Calculate power
    lowpower = sum(abs(X(1:endf)).^2)/endf;
    highpower = sum(abs(X(endf:(Nfft/2)+1)).^2)/((Nfft/2)+1-endf);
    buffer = in(k*(L/2)+1:k*(L/2)+L);
    if abs(highpower/lowpower) > 1.42
        flagi = 0;
        if flagi == 0
            flagi = 1;
            dataplot(iter,:) = [0 k*(1/Fs)*(L/2) k*(1/Fs)*(L/2)+(1/Fs)*(L/2)];
            iter = iter + 1;
        else
            dataplot(iter,:) = [0 k*(1/Fs)*(L/2) k*(1/Fs)*(L/2)+(1/Fs)*(L/2)];
        end
        % Frequency Warping
        for li=1:L/2
            idata (1,1) = buffer(li,1);
            for lw=2:L
                dtmp = idata (1,lw-1) + alfa*wmem(lw);
                idata (1,lw) = wmem(lw) - alfa*dtmp;
                wmem(lw) = dtmp;
            end
        end
        buffer=idata (1,L:-1:1).*win;
        % Overlap add 50%
        channel2(k*(L/2)+1:k*(L/2)+L) = buffer' + channel2(k*(L/2)+1:k*(L/2)+L);
    end
end
% Filtered signal
wavwrite(channel1/max(abs(channel1)),Fs,NBITS,'channel1');
channel2 = bandPassFilter (channel2, Bbp, Abp);
% Output file
```

```
out = channel1 + channel2;
wavwrite(out/(max(abs(out))),Fs,NBITS,'freqwarp');

 flaginit = 0;
flagend = 1;
ptr = 1;
% Find rectangles boundaries
for jj=1:length(dataplot)-1
    if dataplot(jj,:)==0
        break;
    end
    if flaginit == 0 && flagend == 1
        boundaries(ptr) = jj;
        ptr = ptr + 1;
         flaginit = 1;
        flagend = 0;
    end
    if flagend == 0 && flaginit == 1
        if abs(dataplot(jj+1,2)-dataplot(jj,3)) > (1/Fs)*Nfft
            boundaries(ptr) = jj;
            ptr = ptr + 1;
             flaginit = 0;
            flagend = 1;
        end
    end
end
fh1 = figure(1);
set(fh1,'color','white');
% Plot original signal
subplot(3,1,1);
[B,f,t] = spectrogram(in,256,250,Nfft,Fs);
imagesc(t,f,20*log10(abs(B)));
 title('Espectrograma␣Original','FontSize',16);
xlabel('tempo␣(s)','FontSize',14);
ylabel('frequência␣(Hz)','FontSize',14);
set(gca, 'Box', 'off');
axis xy
set(gca, 'TickDir', 'out', 'XTick',0:0.5:length(in)*(1/Fs)-(1/Fs), 'YTick', 0:1000:5500);
lgrays=zeros(100,3);
for i=1:100
    lgrays(i,:) = 1-i/100;
end
colormap(lgrays);
hold on;
% Plot white rectangles
for kk=1:2:length(boundaries)-1
    max = 0;
    min = 100000;
    for ll=boundaries(kk):boundaries(kk+1)
```

```
            if ( dataplot ( ll ,1))*( Fs/Nfft ) > max
                max = ( dataplot ( ll ,1))*( Fs/Nfft );
            end
            if ( dataplot ( ll ,1))*( Fs/Nfft ) < min
                min = ( dataplot ( ll ,1))*( Fs/Nfft );
            end
        end
        plot( dataplot (boundaries(kk ),2):(1/ Fs/Nfft )*(L/2): dataplot (boundaries(kk+1),3),47*( Fs/Nfft ), 'w')
        hold on;
        plot( dataplot (boundaries(kk ),2):(1/ Fs/Nfft )*(L/2): dataplot (boundaries(kk+1),3),129*( Fs/Nfft ), 'w')
        hold on;
        plot( dataplot (boundaries(kk ),2),47*( Fs/Nfft ): Fs/Nfft :129*(Fs/Nfft ), 'w')
        hold on;
        plot( dataplot (boundaries(kk+1),3),47*( Fs/Nfft ): Fs/Nfft :129*(Fs/Nfft ), 'w')
        hold on;
    end
    hold off;
    % Plot filtered signal
    subplot (3,1,2);
    [B,f,t] = spectrogram(channel1 ,250,128, Nfft ,Fs);
    imagesc(t, f,20*log10(abs(B )));
     title ('Espectrograma␣Sinal␣Filtrado␣a␣2kHz','FontSize' ,16);
    xlabel('tempo␣(s)', 'FontSize' ,14);
    ylabel(' frequência␣(Hz)', 'FontSize' ,14);
    set(gca, 'Box', 'off');
    axis xy
    set(gca, 'TickDir', 'out', 'XTick' ,0:0.5: length(in )*(1/Fs)-(1/Fs), 'YTick', 0:1000:5500);
    lgrays=zeros(100,3);
    for i=1:100
        lgrays(i ,:) = 1-i/100;
    end
    colormap(lgrays);
    % Plot filtered signal
    subplot (3,1,3);
    [B,f,t] = spectrogram(out ,250,128, Nfft ,Fs);
    imagesc(t, f,20*log10(abs(B )));
     title ('Espectrograma␣Sinal␣Processado','FontSize' ,16);
    xlabel('tempo␣(s)', 'FontSize' ,14);
    ylabel(' frequência␣(Hz)', 'FontSize' ,14);
    set(gca, 'Box', 'off');
    axis xy
    set(gca, 'TickDir', 'out', 'XTick' ,0:0.5: length(in )*(1/Fs)-(1/Fs), 'YTick', 0:1000:5500);
    lgrays=zeros(100,3);
    for i=1:100
        lgrays(i ,:) = 1-i/100;
    end
    colormap(lgrays);
```

Anexo C

Código Translação ou Compressão

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Algorithm to make the frequency shift or compression %
%                                                      %
% Author : Ricardo Costa                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all;
clear all;
clc;

%Cut-off filter frequencies
Fpass  = 1900; %LouPass Filter
Fpass1 = 1450; %BandPass Filter
Fpass2 = 1950; %BandPass Filter
%Filter attenuation
Apass = 0.1;
Astop = 80;

%FFT length
Nfft = 256;
L = Nfft;

%Window Sine
win = sin((pi/Nfft)*((0:Nfft-1)+0.5));
win = win';

%Read audio signal from wave file
[in, Fs, NBITS] = wavread('teste');

in(129:length(in)+128) = in(1:length(in));
in(1:128) = zeros(128,1);

if mod(length(in), Nfft) ~= 0
    while mod(length(in), Nfft) ~= 0
        in(length(in)+1) = 0;
    end
```

```
end

% Filter coeficients
[Blp, Alp] = lowPassCoefs(Fs, Fpass, Apass, Astop);

% Initialize variables
channel1 = zeros(length(in),1);
channel2 = zeros(length(in),1);
out      = zeros(length(in),1);
X        = zeros(Nfft,1);
newX     = zeros(Nfft,1);
maxl     = zeros(Nfft,1);
step = Fs/Nfft;
initf = round(1500/step);
endf = round(2000/step);
bw = round((endf - initf)/2);
dataplot = [zeros(Nfft,1) zeros(Nfft,1) zeros(Nfft,1)];
frames = length(in)/L;
iter = 1;

% k=0
% Channel 1
% High frequencies filtered
buf = lowPassFilter(in(1:L).*win, Blp, Alp);
channel1(1:L) = buf.*win;
% Channel 2
% Frequency domain
X = fft((in(1:L).*win), Nfft);
%Calculate power spectrum
lowpower = sum(abs(X(1:endf)).^2)/endf;
highpower = sum(abs(X(endf:(Nfft/2)+1)).^2)/((Nfft/2)+1-endf);

if abs(highpower/lowpower) > 1.42
    %Find local peaks and max peaks
    maxl     = zeros(Nfft,1);
    ind = 1;
    maxlocal = 0;
    for bin = 35: 128

        if abs(X(bin)) > maxl(ind)
            if abs(X(bin)) > abs(X(bin+1))
                if abs(X(bin)) > abs(X(bin-1))
                    maxl(ind) = abs(X(bin));
                    ind = ind + 1;
                end
            end
        end

        if abs(X(bin)) > maxlocal
```

```
            maxlocal = abs(X(bin));
            binmax = bin;
        end

    end

    %Search for n important peaks
    n = 0;
    for ii = 1: length(maxl)
        if maxl(ii) > maxlocal*(3/4)
            n = n + 1;
        end
    end
    if n == 1 %Shift
        if mod(binmax,2)==0
            initbin = binmax - bw;
            endbin = binmax + bw;
        else
            initbin = binmax - bw + 1;
            endbin = binmax + bw - 1;
        end
        d = initbin - initf;
        flagi = 0;
        for bin = initbin :endbin
            if flagi == 0
                flagi = 1;
                dataplot(iter,:) = [bin (1/Fs)*(L/2) (1/Fs)*(L/2)+(1/Fs)*(L/2)];
                iter = iter + 1;
            else
                dataplot(iter,:) = [bin (1/Fs)*(L/2) (1/Fs)*(L/2)+(1/Fs)*(L/2)];
            end
            newX(bin-d) = X(bin)*1.0;
        end

    elseif n > 1 %Compression
        flagi = 0;
        for bin=round(2000/step):((Nfft)/2)+1
            fout = uint16((bin*43)/8+1250);
            bout = round(fout/43);
            if ((fout-bout)*43) <= 2
                if mod(bin,2) == 0
                    if mod(bout,2) == 0
                        if flagi == 0
                            flagi = 1;
                            dataplot(iter,:) = [bin 0 (1/Fs)*(L/2)];
                            iter = iter + 1;
                        else
                            dataplot(iter,:) = [bin 0 (1/Fs)*(L/2)];
                        end
```

```
                    newX(bout) = X(bin)*1.0;
                else
                    if flagi == 0
                        flagi = 1;
                        dataplot(iter,:) = [bin 0 (1/Fs)*(L/2)];
                        iter = iter + 1;
                    else
                        dataplot(iter,:) = [bin 0 (1/Fs)*(L/2)];
                    end
                    newX(bout) = X(bin-1)*1.0;
                end
            else
                if mod(bout,2) == 0
                    if flagi == 0
                        flagi = 1;
                        dataplot(iter,:) = [bin 0 (1/Fs)*(L/2)];
                        iter = iter + 1;
                    else
                        dataplot(iter,:) = [bin 0 (1/Fs)*(L/2)];
                    end
                    newX(bout) = X(bin-1)*1.0;
                else
                    if flagi == 0
                        flagi = 1;
                        dataplot(iter,:) = [bin 0 (1/Fs)*(L/2)];
                        iter = iter + 1;
                    else
                        dataplot(iter,:) = [bin 0 (1/Fs)*(L/2)];
                    end
                    newX(bout) = X(bin)*1.0;
                end
            end
        end
    end
else
    newX = zeros(Nfft,1);
end
% Get negative frequencies
newX(Nfft:-1:(Nfft/2)+2) = conj(newX(2:(Nfft/2)));
% Return to time domain
X = real(ifft(newX,Nfft));
newX = zeros(Nfft,1);
channel2(1:L) = X(1:L).*win;

%k>0
for k=1:(2*frames)-2
    % Channel1
    % High frequencies filtered
```

```
buf = lowPassFilter (in(k*(L/2)+1:k*(L/2)+L).*win, Blp, Alp);
% Overlap add 50%
channel1(k*(L/2)+1:k*(L/2)+L) = (buf.*win) + channel1(k*(L/2)+1:k*(L/2)+L);
% Channel2
% Frequency domain
X = fft ((in(k*(L/2)+1:k*(L/2)+L).*win), Nfft );
%Calculate power spectrum
lowpower = sum(abs(X(1:endf )).^2)/ endf;
highpower = sum(abs(X(endf:(Nfft /2)+1)).^2)/(( Nfft/2)+1-endf);

if abs(highpower/lowpower) > 1.42
    %Find local peaks and max peaks
    ind = 1;
    maxl = zeros(Nfft ,1);
    maxlocal = 0;
    for bin = 35: 128

        if abs(X(bin)) > maxl(ind)
            if abs(X(bin)) > abs(X(bin+1))
                if abs(X(bin)) > abs(X(bin-1))
                    maxl(ind) = abs(X(bin));
                    ind = ind + 1;
                end
            end
        end

        if abs(X(bin)) > maxlocal
            maxlocal = abs(X(bin));
            binmax = bin;
        end

    end

    %Search for n important peaks
    n = 0;
    for ii = 1: length(maxl)
        if maxl(ii) > maxlocal*(3/4)
            n = n + 1;
        end
    end
    if n == 1 %Shift
        if mod(binmax,2)==0
            initbin = binmax - bw;
            endbin = binmax + bw;
        else
            initbin = binmax - bw + 1;
            endbin = binmax + bw - 1;
        end
        d = initbin - initf ;
```

```
    flagi = 0;
    for bin = initbin :endbin
        if flagi == 0
            flagi = 1;
            dataplot ( iter ,:) = [bin k*(1/Fs)*(L/2) k*(1/Fs)*(L/2)+(1/Fs)*(L/2)];
            iter = iter + 1;
        else
            dataplot ( iter ,:) = [bin k*(1/Fs)*(L/2) k*(1/Fs)*(L/2)+(1/Fs)*(L/2)];
        end
        newX(bin-d) = X(bin)*1.0;
    end

elseif n > 1 %Compression
    flagi = 0;
    for bin=round(2000/step ):(( Nfft )/2)+1
        fout = uint16 (( bin*43)/8+1250);
        bout = round(fout /43);
        if fout-bout*43 <= 2
            if mod(bin,2) == 0
                if mod(bout,2) == 0
                    if flagi == 0
                        flagi = 1;
                        dataplot ( iter ,:) = [bin k*(1/Fs)*(L/2) k*(1/Fs)*(L/2)+(1/Fs)*(L/2)];
                        iter = iter + 1;
                    else
                        dataplot ( iter ,:) = [bin k*(1/Fs)*(L/2) k*(1/Fs)*(L/2)+(1/Fs)*(L/2)];
                    end
                    newX(bout) = X(bin)*1.0;
                else
                    if flagi == 0
                        flagi = 1;
                        dataplot ( iter ,:) = [bin k*(1/Fs)*(L/2) k*(1/Fs)*(L/2)+(1/Fs)*(L/2)];
                        iter = iter + 1;
                    else
                        dataplot ( iter ,:) = [bin k*(1/Fs)*(L/2) k*(1/Fs)*(L/2)+(1/Fs)*(L/2)];
                    end
                    newX(bout) = X(bin-1)*1.0;
                end
            else
                if mod(bout,2) == 0
                    if flagi == 0
                        flagi = 1;
                        dataplot ( iter ,:) = [bin k*(1/Fs)*(L/2) k*(1/Fs)*(L/2)+(1/Fs)*(L/2)];
                        iter = iter + 1;
                    else
                        dataplot ( iter ,:) = [bin k*(1/Fs)*(L/2) k*(1/Fs)*(L/2)+(1/Fs)*(L/2)];
                    end
                    newX(bout) = X(bin-1)*1.0;
                else
```

```
                        if flagi == 0
                            flagi = 1;
                            dataplot(iter,:) = [bin k*(1/Fs)*(L/2) k*(1/Fs)*(L/2)+(1/Fs)*(L/2)];
                            iter = iter + 1;
                        else
                            dataplot(iter,:) = [bin k*(1/Fs)*(L/2) k*(1/Fs)*(L/2)+(1/Fs)*(L/2)];
                        end
                        newX(bout) = X(bin)*1.0;
                    end
                end
            end
        end
    end
    else
        newX = zeros(Nfft,1);
    end
    % Get negative frequencies
    newX(Nfft:-1:(Nfft/2)+2) = conj(newX(2:(Nfft/2)));
    % Return to time domain
    X = real(ifft(newX,Nfft));
    newX = zeros(Nfft,1);
    %Overlap add 50%
    channel2(k*(L/2)+1:k*(L/2)+L) = (X(1:L).*win) + channel2(k*(L/2)+1:k*(L/2)+L);
end
% Filtered signal
wavwrite(channel1/max(abs(channel1)),Fs,NBITS,'channel1');
% Output file
out = channel1 + channel2;
wavwrite(out/max(abs(out)),Fs,NBITS,'transcomp');

flaginit = 0;
flagend = 1;
ptr = 1;
% Find rectangles boundaries
for jj=1:length(dataplot)-1
    if dataplot(jj,:)==0
        break;
    end
    if flaginit == 0 && flagend == 1
        boundaries(ptr) = jj;
        ptr = ptr + 1;
        flaginit = 1;
        flagend = 0;
    end
    if flagend == 0 && flaginit == 1
        if abs(dataplot(jj+1,2)-dataplot(jj,3)) > (1/Fs)*Nfft
            boundaries(ptr) = jj;
            ptr = ptr + 1;
            flaginit = 0;
```

```
            flagend  = 1;
        end
    end
end
subplot (3,1,1);
% Plot original signal
[B,f,t] = spectrogram(in,256,250,Nfft,Fs);
imagesc(t,f,20*log10(abs(B)));
axis xy
lgrays=zeros(100,3);
for i=1:100
    lgrays(i,:) = 1-i/100;
end
colormap(lgrays);
hold on;
% Plot white rectangles
for kk=1:2:length(boundaries)-1
    max = 0;
    min = 100000;
    for ll=boundaries(kk):boundaries(kk+1)
        if (dataplot(ll,1))*(Fs/Nfft) > max
            max = (dataplot(ll,1))*(Fs/Nfft);
        end
        if (dataplot(ll,1))*(Fs/Nfft) < min
            min = (dataplot(ll,1))*(Fs/Nfft);
        end
    end
    plot(dataplot(boundaries(kk),2):(1/Fs)*(L/2):dataplot(boundaries(kk+1),3),47*(Fs/Nfft),'w')
    hold on;
    plot(dataplot(boundaries(kk),2):(1/Fs)*(L/2):dataplot(boundaries(kk+1),3),129*(Fs/Nfft),'w')
    hold on;
    plot(dataplot(boundaries(kk),2),47*(Fs/Nfft):Fs/Nfft:129*(Fs/Nfft),'w')
    hold on;
    plot(dataplot(boundaries(kk+1),3),47*(Fs/Nfft):Fs/Nfft:129*(Fs/Nfft),'w')
    hold on;
end
hold off;
% Plot filtered signal
subplot (3,1,2);
[B,f,t] = spectrogram(channel1,256,128,Nfft,Fs);
imagesc(t,f,20*log10(abs(B)));
axis xy
lgrays=zeros(100,3);
for i=1:100
    lgrays(i,:) = 1-i/100;
end
colormap(lgrays);
% Plot compressed signal
subplot (3,1,3);
```

```
[B,f,t] = spectrogram(out,256,128,Nfft,Fs);
imagesc(t,f,20*log10(abs(B)));
axis xy
lgrays=zeros(100,3);
for i=1:100
    lgrays(i,:) = 1-i/100;
end
colormap(lgrays);
```

Anexo D

Código Compressão, Translação e Síntese de Harmónicas

```
%-------------------------------------------------------------------------
%
% This MATLAB implements the proof-of-concept of a frequency domain pitch
% shifter
%
% Anibal Ferreira, University of Porto, PORTUGAL / SEEGNAL Research
% April 2nd, 2009
%
%-------------------------------------------------------------------------

%%%%%%%%%%%%
% Adapted for Synthesis of Harmónicas
% by Ricardo Costa
%%%%%%%%%%%%

close all;
clear all;
clc;

% flag indicating there is no frame history

sync=0;

%
% input audio file (raw PCM)
%

 inpfile = 'teste.pcm';
 outfile = 'channel2.pcm';

%
% N         = size of the ODFT and MDCT transforms
```

```
% win       = sine window
%
% N=1024; N2=N/2; N4=N/4;
N=256; N2=N/2; N4=N/4;
win=sin(pi/N*([0:N-1]+0.5));
% FS=32000;
FS = 11025;
order = 18; % order of LPC / cepstrum spectral envelope model

data    = zeros(1,N);     % input data
idata   = zeros(1,N);     % input ( int ) data
odata   = zeros(1,N);     % output ( int ) data
osdata  = zeros(1, N);    % buffered data
tmpdata = zeros(1, N2);
fdata   = zeros(1,N);     % ( float complex) data
ofdata  = zeros(1,N);     % ( float complex) data
ntonaltrack = zeros(1,1000);     % MAX is 1000 frames
tonaltrack  = zeros(1000,300); % MAX is 1000 frame segments x 300 partials in each frame

initf = round(1500/43);
endf = round(2000/43);
bw = round((endf - initf )/2);
[Blp, Alp] = lowPassCoefs(FS, 1900, 0.1, 80);
channel1 = zeros(1,N);
buffer = zeros(1,N/2);
dout = zeros(1,N);
buf = zeros(1,N/2);
L = N;
iter = 1;
iter2 = 1;
newX = zeros(N,1);
dataplot = [];
boundaries = [];

% number of side lobes to be synthesized
sidelobes = 3; % three sidelobes (9 bins per sinusoid in total )


nmaxpartials = 0;

% number of partials that are synthesized on a voiced frame
numpartials=30; % 30 should be sufficient
change = zeros(1, numpartials );
% specify here which partials will have synthetic phase, e.g.
% change(1:20)=1;

decaydB = 10.0; % dB decay of vanishing sinusoids

% OLDPHI retains the phase of the previous ODFT frame (partials only)
```

```
oldphi     = zeros(1,N2);
% OLDMAG retains the magnitude of the previous ODFT frame (partials only)
oldmag     = zeros(1,N2);

% vectores de memória
 oldell    = zeros(1,N2);
 olddeltaell    = zeros(1,N2);


%
% complex vectors for the non-optimal computation of the ODFT, and IODFT transforms
%
direxp  = exp(-i*pi*[0:(N-1)]/N);
invexp  = exp( i*pi*[0:(N-1)]/N);

%-----------------------------------------------------------------------
%
% read audio file
%
 fidr     = fopen( inpfile ,'r' );
fidw     = fopen( outfile ,'w');
 fidch1  = fopen('channel1.pcm','w');
 fidout  = fopen('output.pcm','w');

%
% begin overlap-add
%
[tmpdata, nread] = fread( fidr , N2, 'short' );
data(1,N2+1:N)=tmpdata(1:N2,1).';

k=0; % set frame counter

while(nread==N2),

%
% overlap/add analysis , ODFT
%
    k = k+1; % frame counter

    figure (1); % displays time signal before windowing
    plot ([0:N-1]*1000.0/FS, data(1:N));
    xlabel('Time (ms) \rightarrow');
    ylabel('Amplitude \rightarrow');

    idata=data.*win;

    %%%%%%%%%%%%%%%%
    channel1 = lowPassFilter ( idata , Blp, Alp)';
    channel1 = channel1.*win;
```

```
    fwrite(fidch1, channel1(1:N/2) + buffer, 'short');
    buffer = channel1(N/2+1:N);
    X = fft(idata,N);
    %%%%%%%%%%%%%%%%%

    fdata = idata.*direxp; % this is sub-optimal ODFT computation
    odft=fft(fdata); % this is sub-optimal ODFT computation
    phaseodft = angle(odft(1:N2));

    powerodft = 1E-6+abs(odft).^2;
    envodft=10*log10(powerodft);

%
% looks in the spectrum for the most prominent harmonic structure, if any
%
% NOTE 1: searchtonal() is a compiled (MEX) C function
% NOTE 2: althougth interface is prepared for two harmonic structures, only the first is effectively used
%

    % estranhamente esta rotina searchtonal() tem que ser colocada aqui senão
    % dá erro, parece ser erro do Matlab que não quer uso de gráficos antes
    [ npartials0 missing0 posstop0 f0pitch0 npartials1 missing1 posstop1 f0pitch1 ] = searchtonal(FS, N2, envodft, powerodft); % N

    figure(2);
    plot([0:N2-1]*FS/N, envodft(1:N2),'b'); % displays original spectrum
    xlabel('Frequency␣(Hz)␣\rightarrow');
    Ylabel('Magnitude␣(dB)␣\rightarrow');
    axis([0 N2*FS/N 0 130])

%
% ODFT smooth spectral envelope model based on cepstrum
% ODFT -> ABS -> LOG -> IODFT -> ShortPassLifter -> ODFT
%
    cepstrum=ifft(envodft); % this is sub-optimal IODFT computation
    cepstrum=cepstrum.*invexp; % frequency domain
    limite=order;
    cepstrum(1+limite:N-limite+1)=0; % ideal short-pass lifter
    envceps=cepstrum.*direxp; % this is sub-optimal ODFT computation
    envceps=real(fft(envceps)); % this is sub-optimal ODFT computation

    figure(2);
    hold on;
    plot([0:N2-1]*FS/N, envceps(1:N2),'k');
    hold off;


    edge=0.5; % 5 dB % modficado em 2set07 para aumentar muito a sensibilidade
    maxima = []; minima = [];
```

```
%
% finds all relevant peaks and valleys in the spectrum
%
[maxima, minima] = peaksvalleys(envodft, N2, edge, edge);

%
% find delta ell of all maxima
%
 deltaell = [];
 if (length(maxima) >1) deltaell = getdeltaell2 (powerodft, maxima); end


disp('Number␣of␣partials␣of␣the␣harmonic␣structure');
 npartials0
disp('Number␣of␣missing␣partials␣in␣the␣harmonic␣structure');
missing0
 if (missing0>0)
     disp(' Position ␣of␣ first ␣missing␣ partial ' );
     posstop0
end
disp('PITCH␣of␣the␣harmonic␣structure');
f0pitch0

 if ( npartials0 >0)
     nmaxpartials = floor ((N2−sidelobes)/f0pitch0 ); % maximum number of partials in the full spectrum
 else
     nmaxpartials = 0;
end

%
% now identify the natural partials that are closest to the ideal harmonic
% alignement ** this is based on maxima() and deltaell () ****
%

 trueell = []; truedeltaell = []; truemag = []; truephi = [];
 if ( npartials0 >0)
     truepeaks = min(nmaxpartials, min(length(maxima)−2,numpartials)); % 30 should be sufficient
 else
     truepeaks = 0;
end


% esta rotina tem o problema de poder não encontrar facilmente o
% verdadeiro pico quando ele está mergulhado em ruído, como tornar mais
% robusto ?
 pointer=1;
 for m=1:truepeaks,
     dtmp = m * f0pitch0 ;
     % tenta colocar pointer no parcial real à direita do parcial ideal
```

```
        % aqui em searchtonal pode-se aproveitar o vector dos parciais
        % já pesquisados
        while ( pointer < length(maxima) && maxima(pointer)-1+deltaell(pointer) <= dtmp),
            pointer=pointer+1;
        end

        difaft = abs(maxima(pointer)-1+deltaell(pointer)-dtmp);
        if (pointer>1)
            difbef = abs(dtmp-maxima(pointer-1)+1-deltaell(pointer-1)); % HAVIA ERRO !
        else
            difbef = 1E4; % just a large number
        end
        if ( difaft < difbef )
            trueell(m) = maxima(pointer);
            truedeltaell(m) = deltaell(pointer);
%             pointer = pointer + 1;
        else
            trueell(m) = maxima(pointer-1);
            truedeltaell(m) = deltaell(pointer-1);
        end
    end

    % fase diferencial
    difphi = zeros(200,2);

    if (truepeaks>0)
        truephi = getphase2(phaseodft, trueell, N);
        truemag = getPSD(envodft, trueell, truedeltaell, N);

        % só se calcula o diferencial de fase para antes e depois, será
        % necessário para cacular para os 4 bins antes e 4 bins depois ?
        % NAO!, induz maior erro nos bins mais laterais
        difphi = difphase2(phaseodft, trueell, N);

        figure(2);
        hold on;
        plot( (trueell-1)*FS/N, 1+envodft(trueell), 'rv' ); % identifies peaks graphically
        hold off;
    end

    truedata = zeros(1,N); % make sure we start with an empty buffer
    if (truepeaks>0)
        if (sync==1) % there is history
            for s=1:truepeaks, % for all tonals in the current frame
                if (change(s)==1)
                    dtmp=0.5*(oldell(s)+olddeltaell(s)+trueell(s)+truedeltaell(s))-1.0;
                    tmpphi=mainarg(oldphi(s)+2*pi/N*dtmp*N/2); % to facilitate readability
                else
                    tmpphi=truephi(s);
```

```
                end
                tmpdata = syntonal2( truedata , trueell (s), truedeltaell (s), truemag(s), tmpphi, difphi(s,:), N, sidelobes );
                truedata = tmpdata;
                oldphi(s)=tmpphi;
                oldmag(s)=truemag(s);
                oldell (s)= trueell (s);
                olddeltaell (s)= truedeltaell (s);
            end
        else %there is no history
            for s=1:truepeaks, % for all tonals in the current frame
                tmpphi=truephi(s); % we may reset this value at sinusoidal birth
                tmpdata = syntonal2( truedata , trueell (s), truedeltaell (s), truemag(s), tmpphi, difphi(s,:), N, sidelobes );
                truedata = tmpdata;
                oldphi(s)=tmpphi;
                oldmag(s)=truemag(s);
                oldell (s)= trueell (s);
                olddeltaell (s)= truedeltaell (s);
            end
        end
        sync=1;
        sussurro = odft - truedata; % sinal diferença (i.e. ruído)
        figure (2);
        hold on
        plot ([0:N2-1]*FS/N, 20*log10(abs(sussurro(1:N2))),'g'); % displays residual spectrum
        axis([0 N2*FS/N 0 130])
        hold off

    else
        sync=0;
        truedata = odft;
        oldphi   = zeros(1,N2);
        oldmag   = zeros(1,N2);
        oldell   = zeros(1,N2);
        olddeltaell   = zeros(1,N2);
    end

    %
    % output data, pick one: the signal or the residual
    %
    fdata(1:N2) = truedata(1:N2); % este é o sintetizado harmónico
%     fdata(1:N2) = sussurro(1:N2); % este é o sinal resíduo


%
% For visualization purposes
%
%
    graphell = []; graphdeltaell = [];
    ntonaltrack(k) = truepeaks; % stores number of spectral peaks found in current frame
```

```
    for m=1:truepeaks,
        graphell(m) = trueell(m);
        graphdeltaell(m) = truedeltaell(m);
    end

%-------------------------------------------------------------------------
% Depicts detected partials
%-------------------------------------------------------------------------

    if (ntonaltrack(k) > 0) % if there are any relevant maxima
        tonaltrack(k,1:ntonaltrack(k)) = (graphell + graphdeltaell - 1.0); % stores accurate bin frequency of each peak
        figure(3);
        hold on;
        plot(k, tonaltrack(k,1:ntonaltrack(k)), '*'); % depicts all tonals found in current frame
        axis([0 k+1 0 200]) % 200 is arbitrary, only chosen to represent low frequency tonals
        xlabel('Time (frames) \rightarrow');
        Ylabel('Frequency (accurate bin scale) \rightarrow');
        hold off;
    end

%
% IODFT, overlap/add reconstruction
%
    % Consonant Power
    lowpower = sum(abs(X(1:round(2000/43))))/round(2000/43);
    highpower = sum(abs(X(round(2000/43):(N/2)+1)))/((N/2)+1-round(2000/43));

    % Vowel /e/, /i/ power
    lowpowerv = sum(abs(X(1:round(1000/43))))/round(1000/43));
    midpowerv = sum(abs(X(round(1000/43):round(2000/43))))/(round(2000/43)-round(1000/43));
    highpowerv = sum(abs(X(round(2000/43):93))/(93-round(2000/43)));

    if (highpower/lowpower) > 1.42
       %Procurar os maximos locais e maximo absoluto
        ind = 1;
        maxl = zeros(N,1);
        maxlocal = 0;
        for bin = 35: 128
            if abs(X(bin)) > maxl(ind)
                if abs(X(bin)) > abs(X(bin+1))
                    if abs(X(bin)) > abs(X(bin-1))
                        maxl(ind) = abs(X(bin));
                        ind = ind + 1;
                    end
                end
            end
            if abs(X(bin)) > maxlocal
                maxlocal = abs(X(bin));
                binmax = bin;
```

```
    end
end

%Processar maximos locais para escolher processamento
n = 0;
for ii = 1: length(maxl)
    if maxl(ii) > maxlocal*(3/4)
        n = n + 1;
    end
end
if n == 1 % Translacao
    if mod(binmax,2)==0
        initbin = binmax - bw;
        endbin = binmax + bw;
    else
        initbin = binmax - bw + 1;
        endbin = binmax + bw - 1;
    end
    d = initbin - initf ;
    flagi = 0;
    for bin = initbin :endbin
        if flagi == 0
            flagi = 1;
            dataplot ( iter ,:) = [bin k*(1/FS)*(L/2) k*(1/FS)*(L/2)+(1/FS)*(L/2)];
            iter = iter + 1;
        else
            dataplot ( iter ,:) = [bin k*(1/FS)*(L/2) k*(1/FS)*(L/2)+(1/FS)*(L/2)];
        end
        newX(bin-d) = X(bin)*1.4;
    end
elseif n > 1 % Compressao
    flagi = 0;
    for bin=round(2000/43):(N/2)+1
        fout = uint16 ((bin*43)*.2286+742.85);
        bout = round(fout/43);
        if fout-bout*43 <= 2
            if mod(bin,2) == 0
                if mod(bout,2) == 0
                    if flagi == 0
                        flagi = 1;
                        dataplot ( iter ,:) = [bin k*(1/FS)*(L/2) k*(1/FS)*(L/2)+(1/FS)*(L/2)];
                        iter = iter + 1;
                    else
                        dataplot ( iter ,:) = [bin k*(1/FS)*(L/2) k*(1/FS)*(L/2)+(1/FS)*(L/2)];
                    end
                    newX(bout) = X(bin)*1.4;
                else
                    if flagi == 0
                        flagi = 1;
```

```
                    dataplot ( iter ,:) = [bin k∗(1/FS)∗(L/2) k∗(1/FS)∗(L/2)+(1/FS)∗(L/2)];
                    iter = iter + 1;
                else
                    dataplot ( iter ,:) = [bin k∗(1/FS)∗(L/2) k∗(1/FS)∗(L/2)+(1/FS)∗(L/2)];
                end
                newX(bout) = X(bin−1)∗1.4;
            end
        else
            if mod(bout,2) == 0
                if flagi == 0
                    flagi = 1;
                    dataplot ( iter ,:) = [bin k∗(1/FS)∗(L/2) k∗(1/FS)∗(L/2)+(1/FS)∗(L/2)];
                    iter = iter + 1;
                else
                    dataplot ( iter ,:) = [bin k∗(1/FS)∗(L/2) k∗(1/FS)∗(L/2)+(1/FS)∗(L/2)];
                end
                newX(bout) = X(bin−1)∗1.4;
            else
                if flagi == 0
                    flagi = 1;
                    dataplot ( iter ,:) = [bin k∗(1/FS)∗(L/2) k∗(1/FS)∗(L/2)+(1/FS)∗(L/2)];
                    iter = iter + 1;
                else
                    dataplot ( iter ,:) = [bin k∗(1/FS)∗(L/2) k∗(1/FS)∗(L/2)+(1/FS)∗(L/2)];
                end
                newX(bout) = X(bin)∗1.4;
            end
        end
    end
end
elseif highpowerv/midpowerv > 3.1 && lowpowerv/highpowerv > 2.1
    if isempty( trueell )==0
        maxi = 0;
        for ii=1:length( trueell )
            if truemag( ii ) > maxi && trueell( ii ) > 47
                maxi = truemag( ii );
                binmaxi = ii ;
            end
        end
        d = 36;
        truedata = zeros(1,N);
        for s=binmaxi−1:binmaxi+1 % for all tonals in the current frame
            tmpphi=truephi(s); % we may reset this value at sinusoidal birth
            tmpdata = syntonal2( truedata , d, truedeltaell (s), truemag(s), tmpphi, difphi(s ,:), N, sidelobes );
            newX = tmpdata';
            truedata = tmpdata;
            oldphi(s)=tmpphi;
            oldmag(s)=truemag(s);
```

```
            oldell(s)= trueell(s);
            olddeltaell(s)= truedeltaell(s);
            d = d + 4;
        end
        dataplot2(iter2,:) = [0 k*(1/FS)*(L/2) k*(1/FS)*(L/2)+(1/FS)*(L/2)];
        iter2 = iter2 + 1;
    else
        newX = zeros(N,1);
    end

elseif highpowerv/midpowerv > 2 && lowpowerv/highpowerv > 1
    if isempty(trueell)==0
        maxi = 0;
        for ii=1:length(trueell)
            if truemag(ii) > maxi && trueell(ii) > 47
                maxi = truemag(ii);
                binmaxi = ii;
            end
        end
        d = 36;
        truedata = zeros(1,N);
        for s=binmaxi-1:binmaxi+1 % for all tonals in the current frame
            tmpphi=truephi(s); % we may reset this value at sinusoidal birth
            tmpdata = syntonal2(truedata, d, truedeltaell(s), truemag(s), tmpphi, difphi(s,:), N, sidelobes);
            newX = tmpdata';
            truedata = tmpdata;
            oldphi(s)=tmpphi;
            oldmag(s)=truemag(s);
            oldell(s)= trueell(s);
            olddeltaell(s)= truedeltaell(s);
            d = d + 4;
        end
        bin = 1;
        dataplot2(iter2,:) = [bin k*(1/FS)*(L/2) k*(1/FS)*(L/2)+(1/FS)*(L/2)];
        iter2 = iter2 + 1;
    else
        newX = zeros(N,1);
    end
else
    newX = zeros(N,1);
end
newX(N:-1:(N/2)+2) = conj(newX(2:(N/2)));
X = real(ifft(newX,N));
dout = X'.*win;
fwrite(fidw, dout(1:N/2)+buf, 'short');
buf = dout(N/2+1:N);
newX = zeros(N,1);
%Fim da Alteracao!
```

```
    fdata(N:-1:N2+1)=conj(fdata(1:N2));
    ofdata=ifft(fdata);          % this is sub-optimal IODFT computation
    ofdata=ofdata.*invexp;       % this is sub-optimal IODFT computation
    odata=real(ofdata);
    odata=odata.*win;

    tmpdata(1,1:N2)=floor(0.5+osdata(1,1:N2)+odata(1:N2));
    osdata=odata(N2+1:N);

%   fwrite(fidw, tmpdata(1,1:N2), 'short');

    data(1,1:N2)=data(1,1+N2:N);

    [tmpdata, nread]=fread(fidr, N2, 'short'); % reads new half-segment
    if nread<N2,
       tmpdata(nread+1:N2,1)=zeros(N2-nread,1);
    end
    data(1,N2+1:N)=tmpdata(1:N2,1).';

%   pause;
end
fclose(fidr);
fclose(fidw);
fclose(fidch1);

fidch2 = fopen(outfile,'r');
fidch1 = fopen('channel1.pcm','r');

[ch1data, ndata]=fread(fidch1, N2, 'short');
[ch2data, ndat]=fread(fidch2, N2, 'short');

while ndata ~= 0
    fwrite(fidout, ch1data+ch2data, 'short');
    [ch1data, ndata]=fread(fidch1, N2, 'short');
    [ch2data, ndat]=fread(fidch2, N2, 'short');
end

fclose(fidch2);
fclose(fidch1);
fclose(fidout);

disp('END of processing!');

fidin = fopen(inpfile,'r');
fidch1 = fopen('channel1.pcm','r');
fidout = fopen('output.pcm','r');

n = 1;
k = 0;
```

```
while n ~= 0
    [tmp n] = fread( fidin ,1024,'short');
    datain(k+1:k+n) = tmp;
    k = k + n;
end

n = 1;
k = 0;
while n ~= 0
    [tmp n] = fread( fidch1 ,1024,'short');
    datach1(k+1:k+n) = tmp;
    k = k + n;
end

n = 1;
k = 0;
while n ~= 0
    [tmp n] = fread( fidout ,1024,'short');
    dataout(k+1:k+n) = tmp;
    k = k + n;
end

 flaginit = 0;
flagend = 1;
ptr = 1;
if ~isempty(dataplot)
    for jj=1:length(dataplot)-1
        if dataplot(jj,:)==0
            break;
        end
        if flaginit == 0 && flagend == 1
            boundaries(ptr) = jj ;
            ptr = ptr + 1;
             flaginit = 1;
            flagend = 0;
        end
        if flagend == 0 && flaginit == 1
            if abs(dataplot(jj+1,2)-dataplot(jj,3)) > (1/FS)*N
                boundaries(ptr) = jj ;
                ptr = ptr + 1;
                 flaginit = 0;
                flagend = 1;
            end
        end
    end
end

 flaginit2 = 0;
flagend2 = 1;
```

```
ptr2 = 1;
for jj=1:length(dataplot2)-1
    if dataplot2(jj,:)==0
        break;
    end
    if flaginit2 == 0 && flagend2 == 1
        boundaries2(ptr2) = jj;
        ptr2 = ptr2 + 1;
        flaginit2 = 1;
        flagend2 = 0;
    elseif flagend2 == 0 && flaginit2 == 1
        if abs(dataplot2(jj+1,2)-dataplot2(jj,3)) > (1/FS)*N
            boundaries2(ptr2) = jj;
            ptr2 = ptr2 + 1;
            flaginit2 = 0;
            flagend2 = 1;
        end
    end
end

fh1 = figure(1);
set(fh1,'color','white');
subplot(3,1,1);
[B,f,t] = spectrogram(datain,256,250,N,FS);
imagesc(t,f,20*log10(abs(B)));
title('Espectrograma␣Original','FontSize',16);
xlabel('tempo␣(s)','FontSize',14);
ylabel('frequência␣(Hz)','FontSize',14);
set(gca, 'Box', 'off');
axis xy
set(gca, 'TickDir', 'out', 'XTick',0:0.2:length(datain)*(1/FS)-(1/FS), 'YTick', 0:1000:5500);
lgrays=zeros(100,3);
for i=1:100
    lgrays(i,:) = 1-i/100;
end
colormap(lgrays);
hold on;
% Plot White rectangles for consonants
if ~isempty(boundaries)
    for kk=1:2:length(boundaries)-1
        max = 0;
        min = 100000;
        for ll=boundaries(kk):boundaries(kk+1)
            if (dataplot(ll,1))*(FS/N) > max
                max = (dataplot(ll,1))*(FS/N);
            end
            if (dataplot(ll,1))*(FS/N) < min
                min = (dataplot(ll,1))*(FS/N);
            end
```

```
        end
        plot(dataplot(boundaries(kk),2):(1/FS)*(L/2):dataplot(boundaries(kk+1),3),47*(FS/N),'w')
        hold on;
        plot(dataplot(boundaries(kk),2):(1/FS)*(L/2):dataplot(boundaries(kk+1),3),129*(FS/N),'w')
        hold on;
        plot(dataplot(boundaries(kk),2),47*(FS/N):FS/N:129*(FS/N),'w')
        hold on;
        plot(dataplot(boundaries(kk+1),3),47*(FS/N):FS/N:129*(FS/N),'w')
        hold on;
    end
end

%Plot vowels /e/ /i/
for kk=1:2:length(boundaries2)-1
    max2 = 0;
    min2 = 100000;
    for ll=boundaries2(kk):boundaries2(kk+1)
        if (dataplot2(ll,1))*(FS/N) > max2
            max2 = (dataplot2(ll,1))*(FS/N);
        end
        if (dataplot2(ll,1))*(FS/N) < min2
            min2 = (dataplot2(ll,1))*(FS/N);
        end
    end
    plot(dataplot2(boundaries2(kk),2):(1/FS)*(L/2):dataplot2(boundaries2(kk+1),3),47*(FS/N),'w')
    hold on;
    plot(dataplot2(boundaries2(kk),2):(1/FS)*(L/2):dataplot2(boundaries2(kk+1),3),129*(FS/N),'w')
    hold on;
    plot(dataplot2(boundaries2(kk),2),47*(FS/N):FS/N:129*(FS/N),'w')
    hold on;
    plot(dataplot2(boundaries2(kk+1),3),47*(FS/N):FS/N:129*(FS/N),'w')
    hold on;
end
hold off;

subplot(3,1,2);
[B,f,t] = spectrogram(datach1,256,250,N,FS);
imagesc(t,f,20*log10(abs(B)));
title('Espectrograma␣Sinal␣Filtrado␣a␣2kHz','FontSize',16);
xlabel('tempo␣(s)','FontSize',14);
ylabel('frequência␣(Hz)','FontSize',14);
set(gca, 'Box', 'off');
axis xy
set(gca, 'TickDir', 'out', 'XTick',0:0.2:length(datain)*(1/FS)-(1/FS), 'YTick', 0:1000:5500);
lgrays=zeros(100,3);
for i=1:100
    lgrays(i,:) = 1-i/100;
end
colormap(lgrays);
```

```
subplot (3,1,3);
[B,f,t] = spectrogram(dataout,256,250,N,FS);
imagesc(t,f,20*log10(abs(B)));
 title ('Espectrograma Sinal Processado','FontSize',16);
xlabel('tempo (s)','FontSize',14);
ylabel('frequência (Hz)','FontSize',14);
set(gca, 'Box', 'off');
axis xy
set(gca, 'TickDir', 'out', 'XTick',0:0.2:length(datain)*(1/FS)-(1/FS), 'YTick', 0:1000:5500);
lgrays=zeros(100,3);
for i=1:100
    lgrays(i,:) = 1-i/100;
end
colormap(lgrays);

fclose(fidout);
fclose(fidch1);
fclose(fidin);
```

Referências

Aníbal Ferreira, Isabel Trancoso, Fernando Pereira e Carlos Salema. *Comunicações Audiovisuais - Tecnologias, Normas e Aplicações*. IST Press, 2009.

Aníbal J. S. Ferreira. Utilização de sistema passa-tudo para realizar um mapeamento não-linear de frequência. 2007.

Aníbal J. S. Ferreira. Accurate estimation in the domain of the frquency, phase and magnitude of stationary sinusoids. *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 47–50, 2001.

F. J. Fraga, L. P. C. S. Prates e M. C. M. Iorio. Frequency compression/transposition of fricatives consonants for the hearing impaired with high-frequency dead regions. *Proceedings of the 9th Annual Conference of the International Speech Communication Association (Interspeech 2008) incorporating SST'08*, vol.01:2238–2241, 2008.

Aki Härmä, Matti Karjalainen, Lauri Savioja, Vesa Välimäki, Unto K. Laine e Jyri Huopaniemi. Frequency-warped signal processing for audio applications. *Journal Audio Engineering Society*, vol.48:1011 – 1031, 2000.

Maria Helena Mira Mateus, Isabel Falé e Maria João Freitas. *Fonética e Fonologia do Português*. Universidade aberta, 2005.

C. M. Aguilera Muñoz, Peggy B. Nelson, Janet C. Rutledge e A. Gago. Frequency lowering processing for listeners with significant hearing loss. *IEEE*, 1999.

Larry D. Paarmann. Nonlinear spectrum compression for the hearing impaired via a frequency-domain processing algorithm. *Proceedings of the 28th IEEE EMBS Annual International Conference*, 2006.

Larry D. Paarmann e Michael D. Guiher. A nonlinear spectrum compression algorithm for the hearing impaired. *IEEE*, 1989.

Matthew H. Power. Recognition processing systems for speech enhancement : application to frequency lowering. *IEEE*, 1989.

Charlotte M. Reed, Bruce L. Hicks, Louls D. Braida, e Nathaniel I. Durlac. Discrimination of speech processed by low-pass filtering and pitch-invariant frequency lowering. *Journal Acoustical Society of America*, vol.74, 1983.

Charlotte M. Reed, Kenneth I. Schultz, Louis D. Braida e Nathaniel I. Durlac. Discrimination and identification of frequency-lowered speech in listeners with high-frequency hearing impairment. *Journal Acoustical Society of America*, vol.78, 1985.

Joanna D. Robinson, Thomas Baer e Brian C.J. Moore. Using transposition to improve consonant discrimination and detection for listeners with severe high-frequency hearing loss. *International Journal of Audiology*, vol.46:pages 293 – 308, 2007.

Thomas D. Rossing. *Handbook of Acoustics*. Springer, 2007.

Andrea Simpson, Adam A. Hersbach e Hugh J. McDermott. Improvements in speech perception with an experimental nonlinear frequency compression hearing device. *International Journal of Audiology*, vol.44:pages 281 – 292, 2005.

Andrea Simpson, Adam A. Hersbach e Hugh J. McDermott. Frequency compression outcomes in listeners with steeply sloping audiograms. *International Journal of Audiology*, vol.45:619–629, 2006.

Andreas Spanias, Ted Painter e Venkatraman Atti. *Audio Signal Processing and Coding*. Wiley, 2007.

Christopher W. Turner e Richard R. Hurtig. Proportional frequency compression of speech for listeners with sensorineural hearing loss. *Journal Acoustical Society of America*, vol.106, 1999.