Universidade de São Paulo
Instituto de Física de São Carlos
Departamento de Física e Informática

Segmentação de Voz Baseada na Análise
Fractal e na Transformada *Wavelet*

Paulo César Fantinato

São Carlos - SP - Brasil

Outubro de 2008

Universidade de São Paulo
Instituto de Física de São Carlos
Departamento de Física e Informática

Segmentação de Voz Baseada na Análise
Fractal e na Transformada *Wavelet*

Paulo César Fantinato

Dissertação apresentada ao Instituto de Física de São Carlos da Universidade de São Paulo, como parte dos requisitos necessários à obtenção do título de Mestre em Ciências na área de Física Aplicada - Opção Física Computacional.

Orientador: Prof. Dr. Rodrigo Capobianco Guido

São Carlos - SP - Brasil

Outubro de 2008

AUTORIZO A DIVULGAÇÃO TOTAL OU PARCIAL DESTE TRABALHO,
POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS
DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE

Ficha catalográfica elaborada pelo Serviço de Biblioteca e Informação IFSC/USP

Fantinato, Paulo César

Segmentação de voz baseada na análise fractal e na transformada
wavelet. / Paulo César Fantinato; orientador Rodrigo Capobianco Guido –
São Carlos, 2008.

123 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Física - Área
de Concentração: Física Aplicada - opção Computacional) - Instituto de
Física de São Carlos da Universidade de São Paulo.

1. Voz. 2. Fractais. 3. *Wavelets*. I. Título.

FOLHA DE APROVAÇÃO

Paulo César Fantinato

Dissertação apresentada ao Instituto de Física
de São Carlos da Universidade de São Paulo
para obtenção do título de Mestre em Ciências.
Área de Concentração: Física Aplicada – Opção:
Física Computacional.

Aprovado em: 02/12/2008

Comissão Julgadora

Prof. Dr. Rodrigo Capobianco Guido

Instituição: IFSC/USP

Assinatura

Prof. Dr. Mateus Jose Martins

Instituição: IFSC/USP

Assinatura

Prof. Dr. Aledir Silveira Pereira

Instituição: UNESP

Assinatura

“Os ideais são como as estrelas: nunca as alcançaremos. Porém, assim como os marinheiros em alto mar, traçaremos nosso caminho seguindo-as.”

Jean Paul Sartre.

Dedico a presente dissertação a Paula, minha querida esposa, e aos filhos que Deus nos concedeu, Isabela, Marcio e Mariana, alegria de nossas vidas.

Agradecimentos

A Deus, pelo dom da vida e a Jesus, pela presença amiga e certa em todos os momentos de minha vida.

Aos meus pais, Pedro e Leonilda, e também aos meus irmãos e irmãs pelo apoio incondicional.

Aos meus familiares: minha sogra, cunhados e cunhadas, compadres e comadres e sobrinhos e sobrinhas, pelos incentivos recebidos.

Ao meu orientador, pelo suporte no acompanhamento desse trabalho, e ao Instituto de Física de São Carlos, por sediar e prover a infra-estrutura necessária para o desenvolvimento do projeto de pesquisa.

Aos senhores Afonso, Macedo e João, da empresa Scopus Tecnologia Ltda, pelo apoio e concessão do tempo necessário para realização das disciplinas.

Aos meus amigos e companheiros de mestrado, Lucimar, Sylvio, Fabricio, Leonardo, João Paulo, Moura, Mauricio e todos os demais.

Sumário

1	Introdução e Motivação	20
1.0.1	Objetivos	21
1.0.2	Organização do Trabalho	22
2	Revisão Bibliográfica e Direcionamento do Trabalho	23
2.1	Processamento e Segmentação de Sinais de Voz	23
2.1.1	Estudo da fala humana	25
2.1.2	A base de dados TIMIT	28
2.2	Análise Fractal	28
2.2.1	A Dimensão do Fractal	33
2.2.2	Métodos para o Cálculo da Dimensão do Fractal de Sinais Unidimensionais	35
2.3	A Transformada <i>Wavelet</i> Discreta (DWT)	39
2.3.1	Cálculo da DWT	46
2.3.2	Cálculo da DWT inversa (IDWT)	47
2.3.3	Momentos Nulos	47
2.3.4	Famílias de Transformadas <i>Wavelet</i>	47
3	Descrição da Técnica Proposta	50

3.1	A Arquitetura do Sistema Proposto	50
3.2	O Algoritmo Proposto	51
4	Testes e Resultados	58
4.1	Materiais e Métodos	58
4.2	Bateria de Testes	59
5	Conclusões e Trabalhos Futuros	85
	Apêndice I - Coeficientes dos filtros <i>wavelet</i> utilizados nas experiências.	89
	Apêndice II - Código fonte do algoritmo.	108
	Apêndice III - Publicações durante o mestrado.	119
	Referências Bibliográficas	121

Lista de Tabelas

- 2.1 Características das famílias de *wavelets* utilizadas no presente trabalho, incluindo a quantidade de momentos da função *wavelet*. . . 48
- 4.1 Testes do **grupo 1** para os diversos métodos apresentados para um fractal sintetizado, sendo que as implementações estão baseadas em [11] e na técnica proposta. [Resultados para $D = 1.0$]. Legenda: MET: método utilizado ; SW: suporte da *wavelet*, quando for o caso; VO: valor obtido para a dimensão do fractal. Quando MET for o nome de uma *wavelet*, o método corresponde à técnica proposta utilizando a referida *wavelet*. . . 61
- 4.2 Testes do **grupo 1** para os diversos métodos apresentados para um fractal sintetizado, sendo que as implementações estão baseadas em [11] e na técnica proposta. [Resultados para $D = 1.1$]. Legenda: MET: método utilizado ; SW: suporte da *wavelet*, quando for o caso; VO: valor obtido para a dimensão do fractal. Quando MET for o nome de uma *wavelet*, o método corresponde à técnica proposta utilizando a referida *wavelet*. . . 62

- 4.3 Testes do **grupo 1** para os diversos métodos apresentados para um fractal sintetizado, sendo que as implementações estão baseadas em [11] e na técnica proposta. [Resultados para $D = 1.2$]. Legenda: MET: método utilizado ; SW: suporte da *wavelet*, quando for o caso; VO: valor obtido para a dimensão do fractal. Quando MET for o nome de uma *wavelet*, o método corresponde à técnica proposta utilizando a referida *wavelet*. . . 63
- 4.4 Testes do **grupo 1** para os diversos métodos apresentados para um fractal sintetizado, sendo que as implementações estão baseadas em [11] e na técnica proposta. [Resultados para $D = 1.3$]. Legenda: MET: método utilizado ; SW: suporte da *wavelet*, quando for o caso; VO: valor obtido para a dimensão do fractal. Quando MET for o nome de uma *wavelet*, o método corresponde à técnica proposta utilizando a referida *wavelet*. . . 64
- 4.5 Testes do **grupo 1** para os diversos métodos apresentados para um fractal sintetizado, sendo que as implementações estão baseadas em [11] e na técnica proposta. [Resultados para $D = 1.4$]. Legenda: MET: método utilizado ; SW: suporte da *wavelet*, quando for o caso; VO: valor obtido para a dimensão do fractal. Quando MET for o nome de uma *wavelet*, o método corresponde à técnica proposta utilizando a referida *wavelet*. . . 65
- 4.6 Testes do **grupo 1** para os diversos métodos apresentados para um fractal sintetizado, sendo que as implementações estão baseadas em [11] e na técnica proposta. [Resultados para $D = 1.5$]. Legenda: MET: método utilizado ; SW: suporte da *wavelet*, quando for o caso; VO: valor obtido para a dimensão do fractal. Quando MET for o nome de uma *wavelet*, o método corresponde à técnica proposta utilizando a referida *wavelet*. . . 66

- 4.7 Testes do **grupo 1** para os diversos métodos apresentados para um fractal sintetizado, sendo que as implementações estão baseadas em [11] e na técnica proposta. [Resultados para $D = 1.6$]. Legenda: MET: método utilizado ; SW: suporte da *wavelet*, quando for o caso; VO: valor obtido para a dimensão do fractal. Quando MET for o nome de uma *wavelet*, o método corresponde à técnica proposta utilizando a referida *wavelet*. . . 67
- 4.8 Testes do **grupo 1** para os diversos métodos apresentados para um fractal sintetizado, sendo que as implementações estão baseadas em [11] e na técnica proposta. [Resultados para $D = 1.7$]. Legenda: MET: método utilizado ; SW: suporte da *wavelet*, quando for o caso; VO: valor obtido para a dimensão do fractal. Quando MET for o nome de uma *wavelet*, o método corresponde à técnica proposta utilizando a referida *wavelet*. . . 68
- 4.9 Testes do **grupo 1** para os diversos métodos apresentados para um fractal sintetizado, sendo que as implementações estão baseadas em [11] e na técnica proposta. [Resultados para $D = 1.8$]. Legenda: MET: método utilizado ; SW: suporte da *wavelet*, quando for o caso; VO: valor obtido para a dimensão do fractal. Quando MET for o nome de uma *wavelet*, o método corresponde à técnica proposta utilizando a referida *wavelet*. . . 69
- 4.10 Testes do **grupo 1** para os diversos métodos apresentados para um fractal sintetizado, sendo que as implementações estão baseadas em [11] e na técnica proposta. [Resultados para $D = 1.9$]. Legenda: MET: método utilizado ; SW: suporte da *wavelet*, quando for o caso; VO: valor obtido para a dimensão do fractal. Quando MET for o nome de uma *wavelet*, o método corresponde à técnica proposta utilizando a referida *wavelet*. . . 70

4.11	Testes do grupo 1 para os diversos métodos apresentados para um fractal sintetizado, sendo que as implementações estão baseadas em [11] e na técnica proposta. [Resultados para $D = 2.0$]. Legenda: MET: método utilizado ; SW: suporte da <i>wavelet</i> , quando for o caso; VO: valor obtido para a dimensão do fractal. Quando MET for o nome de uma <i>wavelet</i> , o método corresponde à técnica proposta utilizando a referida <i>wavelet</i> . . .	71
4.12	Testes do grupo 2 : sentença e transições entre as palavras, de acordo com a documentação da base TIMIT (TR TIMIT) e de acordo com os testes obtidos com o método proposto (TR MP).	72
4.13	Testes do grupo 2 : sentença e transições entre as palavras, de acordo com a documentação da base TIMIT (TR TIMIT) e de acordo com os testes obtidos com o método proposto (TR MP).	73
4.14	Testes do grupo 2 : sentença e transições entre as palavras, de acordo com a documentação da base TIMIT (TR TIMIT) e de acordo com os testes obtidos com o método proposto (TR MP).	74
4.15	Testes do grupo 2 : sentença e transições entre as palavras, de acordo com a documentação da base TIMIT (TR TIMIT) e de acordo com os testes obtidos com o método proposto (TR MP).	75
4.16	Testes do grupo 2 : sentença e transições entre as palavras, de acordo com a documentação da base TIMIT (TR TIMIT) e de acordo com os testes obtidos com o método proposto (TR MP).	75
4.17	Testes do grupo 3 : alguns fonemas referente a sentença da tabela 4.12 e respectivas posições, de acordo com a documentação da base TIMIT (TR TIMIT) e com os testes obtidos baseados no método proposto (TR MP).	76

4.18	Testes do grupo 3 : alguns fonemas referente a sentença da tabela 4.13 e repectivas posições, de acordo com a documentação da base TIMIT (TR TIMIT) e com os testes obtidos baseados no método proposto (TR MP).	77
4.19	Testes do grupo 3 : alguns fonemas referente a sentença da tabela 4.14 e repectivas posições, de acordo com a documentação da base TIMIT (TR TIMIT) e com os testes obtidos baseados no método proposto (TR MP).	78
4.20	Testes do grupo 3 : alguns fonemas referente a sentença da tabela 4.15 e repectivas posições, de acordo com a documentação da base TIMIT (TR TIMIT) e com os testes obtidos baseados no método proposto (TR MP).	79
4.21	Testes do grupo 3 : alguns fonemas referente a sentença da tabela 4.16 e repectivas posições, de acordo com a documentação da base TIMIT (TR TIMIT) e com os testes obtidos baseados no método proposto (TR MP).	80
4.22	grupo 4 : Transição entre as palavras <i>she</i> e <i>had</i> da frase <i>She had your dark suit in greasy wash water all year</i> de 63488 amostras. Legenda: DWT: base <i>wavelet</i> , SW: suporte correspondente dos filtros, PT: ponto de transição apontado pelo algoritmo proposto. Na TIMIT a transição é indicada na amostra 12783	81
4.23	grupo 4 : Transição entre a palavra <i>year</i> e silêncio da frase <i>She had your dark suit in greasy wash water all year</i> de 63488 amostras. Legenda: DWT: base <i>wavelet</i> , SW: suporte correspondente dos filtros, PT: ponto de transição apontado pelo algoritmo proposto. Na TIMIT a transição é indicada na amostra 60600	82

- 4.24 **grupo 5:** Transição entre fonemas [*sh* → *iy*] da frase *She had your dark suit in greasy wash water all year* de 63488 amostras. Legenda: DWT: base *wavelet*, SW: suporte correspondente dos filtros, PT: ponto de transição apontado pelo algoritmo proposto. Na TIMIT a transição é indicada na amostra **9640** 83
- 4.25 **grupo 5:** Transição entre fonemas [*ih* → *ng*] da frase *She had your dark suit in greasy wash water all year* de 63488 amostras. Legenda: DWT: base *wavelet*, SW: suporte correspondente dos filtros, PT: ponto de transição apontado pelo algoritmo proposto. Na TIMIT a transição é indicada na amostra **24104** 84

Lista de Figuras

2.1	Interpretação física simplificada do sistema bio-gerador de voz [1].	26
2.2	[esquerda]: visão básica do sistema de produção de voz humana; [direita]: detalhe do trato vocal humano e suas sub-partes [1]. . . .	27
2.3	Intervalos no Conjunto de Cantor.	30
2.4	Outro exemplo de um fractal: uma folha de planta e seus detalhes, em menor escala.	31
2.5	Caracterização dos fractais [13].	33
2.6	Construção do triângulo de Sierpinski.	35
2.7	Avaliando o valor de D com o método <i>walking-divider</i>	36
2.8	Obtendo o valor de D com o método <i>box-counting</i>	37
2.9	Ilustração do método <i>prism</i>	37
2.10	Funcionamento da DWT, exemplificado para um sinal $s[\cdot]$ de n amostras discretas e máxima frequência π , decomposto até o ter- ceiro nível. Note o espectro de frequências e a quantidade de amostras presentes em cada sub-banda.	41
2.11	Relação entre os filtros de análise e síntese.	45
2.12	Formato das respostas ao impulso dos filtros <i>wavelet</i> de Haar, Daubechies, Vaidyanathan, Beylkin, Coiflet e Symmlet, respec- tivamente	49

2.13	Formatos das funções <i>scaling</i> dos filtros <i>wavelet</i> de Haar, Daubechies, Vaidyanathan, Beylkin, Coiflet e Symmlet, respectivamente.	49
2.14	Formatos das funções <i>wavelet</i> dos filtros <i>wavelet</i> de Haar, Daubechies, Vaidyanathan, Beylkin, Coiflet e Symmlet, respectivamente.	49
3.1	(acima): exemplo de um sinal contido no vetor $fd[\cdot]$; (abaixo): respectivo envelope.	56
3.2	Arquitetura básica do sistema proposto. Legenda: N.C.T.: não contém transição de fonema ou palavra ; V.N.S.: variação não significativa entre as dimensões dos fractais dos <i>frames</i> consecutivos sob análise ; V.S.: variação significativa entre as dimensões dos fractais dos <i>frames</i> consecutivos sob análise ; T.P.: transição entre palavras ; T.F.: transição entre fonemas ; A.E.S.: alta energia no trecho correspondente do sinal original ; B.E.S.: baixa energia no trecho correspondente do sinal original.	57

Lista de Abreviações

AF	Análise Fractal
ASI	<i>Automatic Speaker Identification</i>
ASR	<i>Automatic Speech Recognition</i>
D	Dimensão do Fractal
DSP	<i>Digital Signal Processing</i>
DTW	<i>Dynamic Time Warping</i>
DWT	<i>Discrete Wavelet Transform</i>
FIR	<i>Finite Impulse Response</i>
FPGA	<i>Field Programmable Gate Array</i>
IDWT	<i>Inverse Discrete Wavelet Transform</i>
LDC	<i>Linguistic Data Consortium</i>
MRA	Análise de Multi-Resolução
PRFB	<i>Perfect Reconstruction Filter Bank</i>
QMF	<i>Quadrature Mirror Filters</i>
WAVE	<i>Waveform Audio Format</i>

Resumo

Ultimamente, a análise fractal (AF) vem sendo utilizada com sucesso na área de processamento digital de voz, especialmente para fins de segmentação de palavras e fonemas, que é uma das etapas fundamentais dos sistemas de reconhecimento automático de fala (*Automatic Speech Recognition* - ASR) e identificação automática de locutor (*Automatic Speaker Identification* - ASI). O uso prático da AF para ASR e ASI depende de dois fatores básicos: baixo custo computacional, para permitir o uso em tempo-real, e precisão nos resultados, para produzir a segmentação correta e entregar dados coerentes à etapa de classificação. Visando atender a esses objetivos, o presente trabalho propõe uma técnica de segmentação de sinais de voz baseada na dimensão do fractal, obtida com o uso da transformada *wavelet* discreta (DWT). Diversas famílias de *wavelets* são testadas e comparadas, sendo que os testes foram realizados com algumas sentenças extraídas da base de dados TIMIT do *Linguistic Data Consortium* (LDC).

Abstract

Nowadays, fractal analysis has been successfully applied to digital speech processing, particularly for words and phonemes segmentation, which represents one of the fundamental steps in automatic speech recognition and speaker identification systems. The practical use of fractal analysis for these purposes should match two principles: low computational cost, to allow use in real-time, and accuracy in the results, to produce a correct segmentation, delivering consistent data to the classifier. Aiming at meeting these two requirements, this work proposes a technique for speech segmentation based on the fractal dimension, obtained by using the discrete wavelet transform (DWT). Many families of wavelets were tested and compared, being the experiments performed with speech data collected from TIMIT corpus provided by the Linguistic Data Consortium.

Capítulo 1

Introdução e Motivação

Em processamento digital de sinais de voz, o reconhecimento automático de fala (*Automatic Speech Recognition* - ASR) e o de locutor (*Automatic Speaker Identification* - ASI) ocupam espaços importantes no desenvolvimento de uma tecnologia que introduz novas linhas de produtos e serviços de comunicação, permitindo estender a capacidade do homem, aumentando sua produtividade e suprimindo suas necessidades sociais. Nesses termos, diversas técnicas têm sido desenvolvidas e aprimoradas a fim de obter resultados satisfatórios com menor custo computacional, aliado ao aumento do poder de processamento dos computadores e microcontroladores. A interação entre o homem e os computadores, eletrodomésticos e equipamentos de telefonia, pela ASR e ASI, tem trazido um conjunto de benefícios sem limites.

Além disso, o uso da Análise Fractal (AF) em processamento digital de sinais (*Digital Signal Processing* - DSP) advém da evolução tecnológica computacional, já que na prática, é extremamente dependente desse recurso. Os estudos iniciais com fractais surgiram da necessidade em expressar formas presentes na

natureza para cuja classificação a geometria clássica de Euclides não fornecia elementos. Seu principal parâmetro é a Dimensão Fractal (D), um valor não inteiro, que expressa a dimensão de cada forma fractal. O valor de D pode ser obtido utilizando técnicas diferentes, propostas durante os anos por estudiosos renomados como Benoit Mandelbrot, Hausdorff-Besicovitch, entre outros. A dimensão fractal, que está diretamente relacionada com a forma do objeto ou sinal em questão, tem valor real entre 1 e 2 para sinais uni-dimensionais, como é o caso do presente trabalho, diferentemente da dimensão Euclidiana que possui somente valores inteiros.

Independentemente do tipo de sistema específico utilizado para ASR ou ASI, assim como do seu princípio básico de funcionamento, que pode ser tanto *pattern-matching* como *knowledge-based* [1], duas fases distintas existirão. A primeira é a fase de segmentação do sinal, em que fonemas ou palavras são isolados, separando-os dos trechos de silêncio. A segunda fase é a de classificação propriamente dita, na qual cada fonema e/ou palavra vai ser reconhecido pelo algoritmo. Este trabalho está diretamente ligado à fase de segmentação, que será baseada no cálculo da dimensão do fractal, obtida, por sua vez, com base na Transformada Wavelet Discreta (*Discrete Wavelet Transform* - DWT).

1.0.1 Objetivos

O objetivo específico do presente trabalho é o de demonstrar a possibilidade do uso da AF para fins de segmentação de palavras e fonemas, com o uso de um algoritmo específico, baseado na DWT, para o cálculo preciso de D , com baixo custo computacional. Para realizar os testes práticos, serão utilizados fonemas extraídos

da base de arquivos de vozes TIMIT do *Linguistic Data Consortium* (LDC).

A técnica proposta dá suporte ao projeto *SpeechAuth*, e também outros projetos em andamento [2], no laboratório *SpeechLab*¹ do Instituto de Física de São Carlos da Universidade de São Paulo (IFSC/USP), e financiado pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP)². Além disso, as contribuições do presente trabalho são intercambiadas com o grupo de pesquisa em processamento de voz da *Microsoft Research* em *Redmond*, WA, USA, da *Shu-Te University*, em Taiwan, R.O.C., e também do INESC, em Lisboa, Portugal.

1.0.2 Organização do Trabalho

Este trabalho está organizado da seguinte forma: o capítulo 2 apresenta uma revisão da literatura, envolvendo os princípios básicos da segmentação de voz, da AF e da DWT. Já o capítulo 3 apresenta, com detalhes, o algoritmo proposto, sendo que os resultados obtidos com os diversos filtros *wavelets* estão descritos no capítulo 4. Tendo em vista os resultados, e apoiado nos conceitos e características estudados durante a revisão de literatura, o capítulo 5 apresenta, de um ponto de vista teórico-prático, as conclusões. Por fim, logo após a lista de referências bibliográficas, três apêndices apresentam, respectivamente, uma lista dos coeficientes dos filtros utilizados nas experiências, o código-fonte da implementação em linguagem de programação de alto nível, assim como as publicações obtidas durante o curso de mestrado do autor.

¹<http://speechlab.ifsc.usp.br>

²<http://www.fapesp.br> - processo nr. 05/0015-1

Capítulo 2

Revisão Bibliográfica e Direcionamento do Trabalho

Neste capítulo, uma revisão da literatura é apresentada, abordando basicamente três tópicos: os conceitos básicos de segmentação de voz, a análise fractal e, por fim, a transformada wavelet discreta. Todos esses conceitos são necessários para a perfeita caracterização do tema proposto na presente dissertação.

2.1 Processamento e Segmentação de Sinais de Voz

Processamento Digital de Voz [3] é uma especialidade que encontra-se em plena expansão. Inúmeras são as aplicações desse ramo da ciência, destacando o reconhecimento automático de fala para fins de interpretação de comandos por máquinas ou robôs, o reconhecimento de locutor para fins de autenticação biométrica, o reconhecimento de patologias no mecanismo de produção da voz para fins biomédicos, entre inúmeras outras tarefas. Independente do objetivo específico, a etapa inicial de um sistema de processamento digital de voz é, como já foi menci-

onado, a segmentação de palavras ou fonemas.

De acordo com [4], o propósito da segmentação de voz, que é também conhecida como alinhamento fonético, segmentação fonética, ou alinhamento texto-fala, é alinhar temporalmente uma sequência de representações textuais fonéticas com um determinado sinal de voz digital, sendo de fato, a identificação dos limites fonéticos na forma de onda do sinal. Considerável pesquisa tem sido concentrada em segmentação de voz, tal como as documentadas em [4][5][6][7], para fins de delimitar palavras e fonemas de uma mesma palavra.

Dentre as técnicas que têm sido desenvolvidas para segmentação de voz, aquelas baseadas em Modelos Ocultos de Markov (*Hidden Markov Models - HMM*), dependentes e independentes do contexto, são bastante tradicionais. *Dynamic Time Warping* (DTW) [8] é uma outra técnica, do tipo *template-matching*, utilizada para segmentação de voz, sem a necessidade de uma etapa de treinamento, sendo também facilmente adaptável para qualquer idioma. Métodos híbridos baseados em redes neurais artificiais e critérios como energia média, seleção de fonemas vozeados e não vozeados, *Mel Frequency Cepstral Coefficients* (MFCCs), métricas espectrais, e outras, são também utilizados. Alguns sistemas atuais também utilizam informações correlacionadas do movimento dos lábios, visando aumentar a eficácia, comprometendo, entretanto, a velocidade na resposta. O algoritmo proposto na presente dissertação possui uma complexidade computacional menor do que os métodos previamente mencionados, mantendo a eficácia.

É bastante importante observar o fato de que transições entre palavras podem

ser mais facilmente determinadas, examinando regiões de baixa energia no sinal de voz, entretanto, nem mesmo existe uma definição única e absoluta para os pontos de transição entre fonemas [4], já que tais transições se desenvolvem de maneira gradual e contínua. Esse fato é facilmente observado na prática, o que faz com que muitas vezes a segmentação “manual” e auditiva seja considerada a mais correta, muito embora essa tarefa seja inviável na prática. No presente trabalho, defende-se o fato de que a análise nas variações da dimensão do fractal de trechos consecutivos do sinal de voz é um critério válido para delimitar palavras e fonemas.

2.1.1 Estudo da fala humana

De acordo com [1], a produção de voz pelo corpo humano, embora seja um mecanismo repleto de detalhes, consiste basicamente da propulsão de ar pelos pulmões, seguida de um processo de filtragem, realizado pelo trato vocal e elementos associados, como ilustram as figuras 2.1 e 2.2. O primeiro detalhe que deve ser observado na figura 2.1 é que as pregas vocais controlam o fluxo de ar fornecido pelos pulmões, fazendo com que esse sinal de excitação seja periódico, vibrando em determinada frequência, ou aperiódico, similar a um sinal ruidoso. Se o sinal for periódico, este período é chamado de período de *pitch* e a voz produzida será classificada como *voiced speech*, caso contrário a voz será classificada como *unvoiced speech*. No primeiro caso encontram-se basicamente as vogais, enquanto no segundo caso estão os demais sons. Dependendo de como agem as estruturas seguintes às pregas vocais, em particular o véu palatino, que controla a passagem do fluxo de ar pelo trato vocal ou nasal, e os próprios tratos vocal e nasal, pode-se ainda refinar essa classificação dos sinais de voz da seguinte forma:

Figura 2.1: Interpretação física simplificada do sistema bio-gerador de voz [1].

- *fricatives*: é um *unvoiced speech* que surge quando há fricção do ar em movimento contra a constrição, causando, em geral, uma turbulência de ar entre a língua e os dentes superiores. Exemplo: th na palavra *thin* da língua Inglesa.
- *plosives*: é um *unvoiced speech* impulsivo, como o t na palavra *top*.
- *whispers*: é um *unvoiced speech* onde uma barreira é criada nas pregas vocais de forma elas permaneçam parcialmente fechadas e sem oscilação, como ocorre quando se pronuncia o h na palavra *he*.
- *voiced fricatives*: são fonemas *voiced*, ou seja de excitação periódica, porém misturado com ruído criado na constrição do trato vocal, atrás dos dentes e contra o palato. Exemplo: z na palavra *zebra*.

Figura 2.2: [esquerda]: visão básica do sistema de produção de voz humana; [direita]: detalhe do trato vocal humano e suas sub-partes [1].

- *unvoiced fricatives*: idem anterior, porém as pregas vocais não vibram simultaneamente com a fricção.
- *voiced plosives*: são fonemas *voiced*, ou seja de excitação periódica, porém misturado com ruído impulsivo criado no trato vocal.
- *unvoiced plosives*: idem anterior, porém as pregas vocais não vibram simultaneamente com o impulso. Exemplo: b na palavra *boat*.

Qualquer palavra ou frase pronunciada por um locutor pode ser dividida em fonemas, cada qual podendo ser classificado como explicado anteriormente.

2.1.2 A base de dados TIMIT

A base de dados de vozes mais comum para realizar testes na área de processamento de sinais de voz é a TIMIT [9], distribuída pelo *Linguistic Data Consortium* (LDC) ¹. A base é composta por sentenças de 420 locutores no conjunto de treinamento e 210 locutores no conjunto de teste, sendo que cada locutor pronuncia 10 sentenças. A base contém arquivos do tipo *Waveform Audio Format* (WAV) [10] amostrados com taxa de 22050 Hz e quantizados com 16 bits, além de arquivos no formato texto que delimitam os pontos de transições entre palavras e fonemas. Em vista dessas características, sentenças da TIMIT foram escolhidas para realizar os testes com o algoritmo proposto no presente trabalho.

2.2 Análise Fractal

A geometria fractal [11] descreve muitas situações que não podem ser explicadas facilmente pela geometria clássica. A origem conceitual dos fractais começou a surgir em função das tentativas de medir o tamanho de objetos para os quais a geometria euclidiana não é apropriada. O matemático Francês Benoit Mandelbrot foi responsável por criar o termo fractal a partir do adjetivo latino *fractus*, do verbo *frangere*, que significa *quebrar*. Um fractal é um objeto geométrico que pode ser dividido em partes, cada uma das quais se parece com o objeto original. Diz-se que os fractais têm infinitos detalhes, são geralmente auto-similares e independem de escala.

Em muitos casos, um fractal pode ser gerado por um padrão repetido, tipicamente

¹<http://www ldc upenn edu/>

um processo recorrente ou iterativo. A couve-flor é um bom exemplo de fractal, já que cada parte dela se parece com a própria couve-flor e cada uma dessas partes é formada por partes ainda menores que também se assemelham com a parte maior, sendo possível pensar nessa divisão infinitamente. Por trás desse exemplo simples está sub-entendido o conceito de escala, isto é, tomando partes maiores da couve-flor observa-se o mesmo tipo de geometria. Em outras palavras, a auto-semelhança é preservada através das escalas.

Ao contrário dos objetos da geometria clássica, fractais não possuem uma dimensão medida no sentido da geometria euclidiana. A noção matemática mais comum sobre dimensão é aquela ligada aos conceitos de espaços vetoriais; esta é uma definição algébrica de dimensão. Sendo assim, o $\mathbb{R}^2$ é gerado por uma base de dois vetores, o $\mathbb{R}^3$ por uma base de 3 vetores, e assim por diante. Com o desenvolvimento da topologia e geometria no começo do século XX, vários matemáticos perceberam que o conceito de dimensão poderia ser diferente do conceito algébrico. Hausdorff foi um dos primeiros a conceber uma nova definição de dimensão. Esta é a que foi posteriormente usada por Mandelbrot para definição de sua geometria fractal, pois a dimensão de Hausdorff tem um forte apelo geométrico.

O matemático russo Georg Ferdinand Ludwig Philip Cantor dedicou a maior parte dos seus estudos ao que atualmente conhecemos como Teoria dos Conjuntos, amplamente difundida, podendo ser considerada como um dos pilares da análise fractal. O termo conhecido como *Conjunto de Cantor*, que constitui um subconjunto fechado dos números reais, foi apresentado pelo autor em 1883. A construção geométrica correspondente leva ao que se conhece como *fractal do terço médio*

de Cantor.

O conjunto de Cantor, aqui denotado por $\mathbb{K}$, é um subconjunto fechado do intervalo $[0; 1] \subset \mathbb{R}$, obtido como complementar de uma reunião de intervalos abertos. Observando a figura 2.3, é possível notar que, para construí-lo, o intervalo $[0; 1]$ deve ser inicialmente dividido em 3 partes iguais. Em seguida, no primeiro nível, retira-se o intervalo $[\frac{1}{3}; \frac{2}{3}]$, ficando com $[0; \frac{1}{3}] \cup [\frac{2}{3}; 1]$, sendo que os pontos $\frac{1}{3}$ e $\frac{2}{3}$ permanecem no conjunto que está sendo construído. No segundo nível, repetem-se o mesmo procedimento para cada intervalo, ou seja, novamente dividem-se os intervalos em três partes iguais e, em seguida, retira-se os respectivos terços médios. E assim sucessivamente nos outros N níveis, tendendo ao infinito.

Figura 2.3: Intervalos no Conjunto de Cantor.

O Conjunto de Cantor é um excelente exemplo que caracteriza a construção dos fractais como sendo processos iterativos. Por isso, não é possível sua representação fiel, visto que tais processos são infinitos. O termo fractal aplica-se, ainda, a construções muito diversas, tanto formas abstratas como formas encontradas na natureza e que são objeto de estudo da física, da geologia, da cristalografia e de

Figura 2.4: Outro exemplo de um fractal: uma folha de planta e seus detalhes, em menor escala.

outras ciências. Um outro exemplo bastante ilustrativo encontra-se na figura 2.4.

Conforme especificou Mandelbrot, em seu primeiro livro sobre o assunto, datado de 1975, deve-se preferir uma definição mais intuitiva do termo fractal, do que uma definição matemática formal. Kenneth Falconer propõe uma definição menos rigorosa, em termos das características das construções ou conjuntos denominados fractais. De acordo com ele, uma dada construção é caracterizada como fractal se possuir todas, ou a maioria, das seguintes características:

- a) possui estrutura fina em qualquer escala: a estrutura fina consiste em um detalhamento infinito. Sucessivas ampliações de um fractal levam a mais e mais detalhes, indefinidamente. Isso não ocorre com as figuras geométricas convencionais, como a circunferência, já que a enorme ampliação de um pequeno arco dela leva a um segmento retilíneo;

- b) não pode ser descrita de maneira simples por uma função analítica ou em linguagem geométrica tradicional: isso se deve ao fato de que o fractal é construído através de processos iterativos, sendo impossível representá-lo por uma função simples;
- c) possui alguma espécie de auto-similaridade ou auto-afinidade, mesmo que estocasticamente: a auto-similaridade, também chamada por Mandelbrot de homotetia interna, consiste em se poder obter réplicas menores do fractal através de sua ampliação. Quando as réplicas são sempre idênticas e obtidas através do mesmo fator de redução, diz-se que o fractal possui auto-similaridade estrita. Já na auto-afinidade, não há mais réplicas, e sim figuras obtidas através de transformações afins. A folha da figura 2.4, por exemplo, possui auto-afinidade estocástica;
- d) sua *dimensão fractal* (D), definida a seguir, é maior que a sua dimensão topológica: a dimensão fractal diz respeito à dimensão espacial, ou seja, ao espaço que o fractal ocupa, existindo diversas formas para sua obtenção;
- e) na maioria dos casos, possui uma lei de formação simples: a lei de formação do fractal é o processo que é repetido a cada iteração. No caso do Conjunto de Cantor, a lei de formação é: divide-se cada segmento em 3 partes, e retira-se o terço médio, como já apresentado.

Por fim, a figura 2.5, obtida em [13], estabelece uma visão mais ampla da caracterização e da natureza dos fractais. Examinando essa figura, percebe-se que os sinais de voz, que são os sinais de interesse no presente trabalho, podem ser caracterizados como sendo fenômenos naturais randômicos, não-

padronizados, estatisticamente auto-similares em diferentes escalas.

Figura 2.5: Caracterização dos fractais [13].

2.2.1 A Dimensão do Fractal

De acordo com a dimensão euclidiana tradicional, a qual denomina-se usualmente dimensão topológica, um ponto possui dimensão 0, uma reta possui dimensão 1, uma superfície possui dimensão 2 e uma porção de espaço possui dimensão 3. Para determinar a dimensão topológica de um objeto, faz-se uma correspondência unívoca desse objeto com um ente geométrico fundamental. Diferentemente, o termo dimensão fractal (D), criado por Mandelbrot, refere-se à dimensão espacial, fazendo referência ao espaço ocupado, ou preenchido, por um sinal ou figura, podendo o valor correspondente ser um número fracionário. A dimensão do

fractal, portanto, consiste de uma medida para o nível de irregularidade ou auto-similaridade do sinal ou figura.

De uma forma relativamente simples, é possível definir D utilizando a seguinte relação:

$$D = \frac{\log(N)}{\log(\frac{1}{T})} \quad (2.1)$$

o que significa que a regra para a formação do fractal consiste em substituir um segmento, parte ou objeto, por N segmentos de tamanho T correspondentes a uma fração do tamanho anterior.

No caso do Conjunto de Cantor, exibido na figura 2.3, fica claro que, a cada etapa, um segmento é substituído por outros 2 segmentos, cada um medindo $\frac{1}{3}$ do anterior. Sendo assim,

$$D = \frac{\log(N)}{\log(\frac{1}{T})} = \frac{\log(2)}{\log(\frac{1}{3})} = \frac{\log(2)}{\log(3)} \approx 0.63 \quad .$$

Um outro exemplo é o da construção do triângulo de Sierpinski [11] que, conforme ilustra a figura 2.6, possui dimensão tal que:

$$D = \frac{\log(N)}{\log(\frac{1}{T})} = \frac{\log(3)}{\log(\frac{1}{4})} = \frac{\log(3)}{\log(4)} \approx 0.79 \quad ,$$

tendo em vista que a cada etapa da construção, um triângulo é substituído por 3 triângulos de área igual a $\frac{1}{4}$ da área do anterior.

Ambos os exemplos acima constituem casos em que a similaridade é facilmente observável e possui caráter não estatístico. Outras classes de sinais ou figuras, como é o caso dos sinais de voz, possuem, conforme já foi mencionado, outro

Figura 2.6: Construção do triângulo de Sierpinski.

tipo de similaridade, de caráter estatístico. Independentemente do tipo de similaridade, os sinais unidimensionais de voz são tais que $1 \leq D \leq 2$, de acordo com [11].

2.2.2 Métodos para o Cálculo da Dimensão do Fractal de Sinais Unidimensionais

Durante o processo de construção de um fractal, com lei de formação bem definida, como nos exemplos acima, é muito simples estabelecer sua dimensão em função dos parâmetros N e T . Entretanto, nos casos em que já se possui o sinal, ou figura, e não fica clara a sua lei de formação por meio de uma simples inspeção visual, um método deve ser utilizado para realizar a medição dessa dimensão. Os métodos mais tradicionais são *walking-divider*, *box-counting*, *prism* e *power spectrum* [11].

O método *walking-divider*, apresentado por Shelberg [11], define a dimensão fractal de um dado sinal como sendo $D = -\beta$, onde β é o coeficiente angular da reta média, obtida via regressão linear por Mínimos Quadrados, que atende ao conjunto de pontos da forma $\log(A_i)$ versus $\log(L_i)$. A variável A_i corresponde ao tamanho do passo, ou comprimento *-step size*, de cada segmento de reta L_i . O

método, que fica mais preciso à medida que o número de pontos cresce, e que está ilustrado através de um exemplo particular na figura 2.7, consiste em reduzir, pela metade, a cada iteração i , o valor de A_i para cobrir a curva, medindo o valor correspondente de L_i . Esse procedimento possui alguns inconvenientes, um dos quais é a definição do valor inicial de A_i , o que pode produzir resultados consideravelmente diferentes.

Figura 2.7: Avaliando o valor de D com o método *walking-divider*.

A técnica de *box-counting*, apresentada por Voss e aperfeiçoada por outros pesquisadores [11], é bastante simples e tradicional, consistindo apenas em obter, como no caso anterior, $D = -\beta$, sendo β também o coeficiente angular da reta média, obtida via regressão linear por Mínimos Quadrados, que atende ao conjunto de pontos da forma $\log(Q_i)$ versus $\log(S_i)$. Nesse caso, Q_i corresponde ao número de quadrados com lados S_i , não sobrepostos, que é necessário para cobrir completamente o fractal. Em geral, o valor inicial para o lado do quadrado, S_0 , corresponde à metade do comprimento ou amplitude ocupado pelo fractal, diminuindo pela metade a cada iteração, até alcançar um valor fracionário relativamente pequeno, dependendo da precisão desejada. A figura 2.8 ilustra um exemplo particular desse método que, em geral, fornece resultados relativamente precisos apenas para $1 \leq D \leq 1.5$, conforme [11]-pp103.

Figura 2.8: Obtendo o valor de D com o método *box-counting*.

O método *prism*, apresentado por Clarke [11], consiste em uma modificação do *box-counting* de forma que $D = 2 - \beta$, sendo β também o coeficiente angular da reta média, obtida via regressão linear por Mínimos Quadrados, que atende ao conjunto de pontos da forma $\log(pa_i)$ versus $\log(ba_i)$. Nesse caso, pa_i corresponde a área total dos 4 lados de um prisma i e ba_i a respectiva área de sua base. Essa definição foi formulada para analisar sinais bidimensionais, ou imagens, sendo que para o caso uni-dimensional, que é o escopo do presente trabalho, os prismas são substituídos por triângulos, conforme ilustra a figura 2.9. Esse método é, em geral, mais preciso do que o anterior, entretanto possui complexidade computacional relativamente maior, tornando-o inviável em determinadas situações.

Figura 2.9: Ilustração do método *prism*.

Diversos outros métodos para a medição de D existem, sendo que a maioria constitui variações das técnicas apresentadas anteriormente. A técnica do *power spectrum*, que entretanto é relativamente diferente das anteriores, é a base do presente trabalho. Essa técnica, em geral baseada na obtenção do espectro de potência do sinal fractal via Transformada de Fourier [3], apresenta resultados relativamente precisos para toda a faixa $1 \leq D \leq 2$, de acordo com estudos apresentados em [11]-pp.110. O algoritmo básico associado ao *power spectrum method* consiste em obter $D = 2 - |H|$, onde $H = \frac{\beta-1}{2}$ é o expoente de Hurst [11], β é o coeficiente angular da reta média, obtida via regressão linear por Mínimos Quadrados, que atende ao conjunto de pontos da forma $\log_2(b_i)$ versus $\log_2(a_i)$, sendo b_i o valor normalizado do espectro de potência da banda i que possui comprimento a_i , para um total de N sub-bandas:

$$\beta = \frac{N \sum_{k=0}^{N-1} \log_2(a_k) \log_2(b_k) - \sum_{k=0}^{N-1} \log_2(b_k) \sum_{k=0}^{N-1} \log_2(a_k)}{N \sum_{k=0}^{N-1} \log_2(a_k)^2 - \left(\sum_{k=0}^{N-1} \log_2(a_k) \right)^2} . \quad (2.2)$$

Em [11], o autor argumenta sobre a necessidade de realizar um pré-processamento no sinal de voz que consiste em uma filtragem do tipo $\frac{1}{k}$ para que seja possível aplicar a técnica *power spectrum*, caso contrário, devido à natureza do sinal de voz, valores fora da faixa $\{1, 2\}$ podem ser obtidos como resultado para D . Esse fato foi facilmente comprovado por intermédio de alguns testes realizados com poucos sinais de vozes coletados de indivíduos diversos.

Tendo em vista a necessidade do pré-processamento, aliado ao fato de que a obtenção do espectro de potência com o uso da Transformada de Fourier é um

processo com complexidade computacional de ordem quadrática em relação ao tamanho do sinal sob análise, ou mesmo logarítmica no caso do uso da Transformada Rápida de Fourier [3], o presente trabalho propõe a obtenção da dimensão do fractal por intermédio de um algoritmo que produz uma representação alternativa para o espectro de potência, baseada na filtragem do sinal de voz via transformada *wavelet*. A técnica proposta, que se encontra descrita com detalhes no próximo capítulo, possui complexidade linear em relação ao sinal de entrada e ainda elimina a necessidade do pré-processamento mencionado anteriormente. Resultados comparativos em termos de custo computacional entre as técnicas mencionadas acima encontram-se detalhadas em [12].

2.3 A Transformada *Wavelet* Discreta (DWT)

A transformada *wavelet* discreta [14] [15] consiste numa alternativa mais eficiente do que a Transformada de Fourier de Tempo Reduzido [3] para realizar a análise tempo-frequência de um sinal, bem como a filtragem e separação em sub-bandas de frequências. A DWT, a qual é objeto fundamental do presente trabalho, funciona como um par de filtros, sendo um deles passa-baixas ($h[\cdot]$) e o outro passa-altas ($g[\cdot]$), em geral, com frequência de corte em metade da máxima frequência presente no sinal de entrada. Dado um sinal discreto, ele é submetido a ambos os filtros via convolução. Cada vez que esse processo é aplicado, diz-se que se tem um nível de decomposição e obtêm-se dois novos sinais, sendo que um deles contém as frequências abaixo da metade da máxima frequência original do sinal e, o outro, contém as frequências acima desse limiar. Em particular, os termos

chamados coeficientes de detalhamento designam o sinal obtido quando da passagem do sinal original pelo filtro passa-altas, e os termos chamados coeficientes de aproximação designam o sinal obtido quando da passagem do sinal original pelo filtro passa-baixas. Após aplicar um nível de decomposição no sinal, apenas o novo sinal obtido pela aplicação do filtro passa-baixas é usado para continuar o processo recursivo de decomposição.

Um detalhe fundamental a ser notado é que, cada vez que um nível da transformação é realizado, os dois novos sinais obtidos são sub-amostrados por 2, pois eles contém apenas metade da faixa de frequências do sinal original, de acordo com o Teorema da Amostragem [3] e como ilustra a figura 2.10. Um sinal de n amostras tem a sua transformada *wavelet* com a mesma quantidade de amostras, sendo composta por uma sequência de coeficientes, iniciando-se com aqueles provenientes da aplicação do filtro passa-baixas no último nível, seguidos pelos coeficientes resultantes da aplicação dos filtros passa-altas nos níveis intermediários, e terminando com os coeficientes resultantes da aplicação do filtro passa-altas do primeiro nível de decomposição. Para realizar a decomposição até o último nível possível, é necessário que o sinal discreto tenha comprimento equivalente a uma potência de 2, sendo possível realizar $\frac{\log(n)}{\log(2)}$ decomposições para um sinal de comprimento n . Um fator muito importante para que um filtro digital seja considerado um filtro *wavelet* é que a resposta em frequência do filtro passa-baixas seja 0 em $\omega = \pi$.

O processo conjunto de filtragem e sub-amostragem por 2, realizado nos sinais da transformada *wavelet* em cada nível, pode ser representado por uma convolução

Figura 2.10: Funcionamento da DWT, exemplificado para um sinal $s[\cdot]$ de n amostras discretas e máxima frequência π , decomposto até o terceiro nível. Note o espectro de frequências e a quantidade de amostras presentes em cada sub-banda.

modificada da seguinte forma:

$$y[\cdot] = x[\cdot] * t[\cdot] = \sum_{k=0}^{n-1} t_k x_{2n-k} \quad , \quad (2.3)$$

ou, mais especificamente:

$$y_{passa-baixas}[\cdot] = x[\cdot] * h[\cdot] = \sum_{k=0}^{n-1} h_k x_{2n-k} \quad (2.4)$$

e

$$y_{passa-altas}[\cdot] = x[\cdot] * g[\cdot] = \sum_{k=0}^{n-1} g_k x_{2n-k} \quad . \quad (2.5)$$

A DWT está diretamente relacionada com a análise de multi-resolução (*Multi-Resolution Analysis* - MRA), proposta por Mallat, Meyer, Stromberg e outros [16] [17], que consiste em decompor um vetor (sinal sob análise) $\vec{f}$ em uma soma de outros vetores pertencentes a uma sequência de sub-espços vetoriais [18]. Em outras palavras, isso significa representar um sinal em vários níveis de resolução. Então, de acordo com a MRA, para um vetor $\vec{f}$ de n pontos tem-se:

$$\vec{f} = \vec{A} + \vec{D} \quad (2.6)$$

onde

$$\vec{A} = \sum_{k=0}^{\frac{n}{2}-1} \langle \vec{f}, \vec{v}_k \rangle \vec{v}_k \quad ,$$

e

$$\vec{D} = \sum_{k=0}^{\frac{n}{2}-1} \langle \vec{f}, \vec{w}_k \rangle \vec{w}_k \quad ,$$

ou seja:

- $\vec{A}$ é a projeção de $\vec{f}$ num sub-espço V , com uma base de $\frac{n}{2}$ vetores;

- $\vec{D}$ é a projeção de $\vec{f}$ num sub-espço W , com uma base de $\frac{n}{2}$ vetores;
- $V \perp W \leftrightarrow \vec{A} \perp \vec{D}$;
- $\vec{v}_i \perp \vec{w}_i \leftrightarrow \langle \vec{v}_i, \vec{w}_i \rangle = 0$.

O processo acima consiste na decomposição em nível 1. Numa transformada *wavelet* de nível 2, o vetor A é novamente decomposto na soma de dois outros vetores ortogonais, podendo esse processo ser repetido $\frac{\log(n)}{\log(2)}$ vezes, conforme já foi mencionado. Dessa forma, generalizando, para uma decomposição de nível j , temos:

$$\vec{f} = \vec{A}_j + \sum_{i=1}^j \vec{D}_i. \quad (2.7)$$

sendo que:

- $\vec{A}_j$ é a projeção de $\vec{f}$ num sub-espço V_j , com uma base contendo $\frac{n}{2^j}$ vetores;
- $\vec{D}_i$ é a projeção de $\vec{f}$ num sub-espço W_i , com uma base contendo $\frac{n}{2^i}$ vetores;
- $V_j \perp W_j \leftrightarrow \vec{A}_j \perp \vec{D}_j$;
- $\vec{v}_{i,j} \perp \vec{w}_{i,j} \leftrightarrow \langle \vec{v}_{i,j}, \vec{w}_{i,j} \rangle = 0$.

O processo anterior equivale a escrever [19]:

$$f[n] = \sum_{k=0}^{\frac{n}{2^j}-1} H_{j,k}[n] \phi_{j,k}[n] + \sum_{t=1}^j \sum_{k=0}^{\frac{n}{2^t}-1} G_{t,k}[n] \psi_{t,k}[n] \quad (2.8)$$

onde

- $\phi[n]$ e $\psi[n]$ formam uma base de Riesz [19] para escrever $\vec{f}$;
- $\phi[n] = \sum_k h_n \phi[2n - k]$, definida recursivamente por dilatações e translações de si mesma é chamada função *scaling* [19];

- $\psi[n] = \sum_k g_n \phi[2n - k]$, também definida recursivamente, é chamada função *wavelet* e é ortogonal a função *scaling*;
- $H_{j,k}[n] = \langle f, \phi_{j,k}[n] \rangle$;
- $G_{t,k}[n] = \langle f, \psi_{t,k}[n] \rangle$;
- $\{0\} \leftarrow \dots \subset V_{-1} \subset V_0 \subset V_1 \subset \dots \rightarrow L^2$;
- se $f[n] \in V_j \rightarrow f[2n] \in V_{j+1}$;
- $V_{j+1} = V_j \oplus W_j$;
- os coeficientes h_k correspondem ao filtro passa-baixas;
- os coeficientes g_k correspondem ao filtro passa-altas;
- $h[\cdot]$ e $g[\cdot]$, que são chamados filtros de análise, formam um par de *Quadrature Mirror Filters* - QMF;
- um filtro com k coeficientes é dito filtro de suporte k .

Cada par de filtros de análise, $h[\cdot]$ e $g[\cdot]$, possui uma única função *scaling* (ϕ) e uma única função *wavelet* (ψ) associadas. A forma de obtenção dessas funções a partir dos filtros, e vice-versa, está documentada com detalhes em [15], não sendo apresentada aqui por estar fora do escopo do trabalho.

É muito importante também o fato de $h[\cdot]$ e $g[\cdot]$ possuírem outros filtros associados, chamados filtros de síntese, representados por $\bar{h}[\cdot]$ e $\bar{g}[\cdot]$, que são utilizados para inverter a transformada, recuperando o sinal original a partir do transformado. Tais filtros obedecem às relações das equações 2.9, 2.10 e 2.11, para

$k = 0, \dots, n - 1$, que ficam mais claras através do exemplo na figura 2.11, para filtros de suporte 4.

$$g_k = (-1)^k h_{N-k-1} \quad , \quad (2.9)$$

$$\bar{h}_k = h_{N-k-1} \quad , \quad (2.10)$$

$$\bar{g}_k = (-1)^{k+1} h_k \quad . \quad (2.11)$$

Quando $h[\cdot]$, $g[\cdot]$, $\bar{h}[\cdot]$, e $\bar{g}[\cdot]$ mantêm as relações acima, eles constituem um banco de filtros de reconstrução perfeita (*perfect reconstruction filter bank* - PRFB) [16] [17], ou seja, as condições de *anti-aliasing* e *no-distortion*, no domínio Z , representadas nas equações 2.12 e 2.13, respectivamente, são satisfeitas. No presente trabalho a inversão da DWT não se faz necessária, entretanto, é desejável que o algoritmo proposto utilize apenas PRFBs. Isso deve-se ao fato de que o algoritmo pode ser embutido em um sistema de ASR ou ASI com funções mais diversificadas que necessite, em algum momento, da inversão da DWT.

Figura 2.11: Relação entre os filtros de análise e síntese.

$$\bar{H}[z] = G[-z] \quad , \quad \bar{G}[z] = -H[-z] \quad . \quad (2.12)$$

$$\bar{H}[z]H[z] + \bar{G}[z]G[z] = 2z^{-N+1} \quad . \quad (2.13)$$

2.3.1 Cálculo da DWT

Para o cálculo da DWT de um sinal, aplica-se o algoritmo de Mallat, que está minunciosamente descrito em [15]. É importante observar que apenas os filtros $h[\cdot]$ e $g[\cdot]$ são utilizados, não sendo necessária a utilização de ϕ e ψ . O procedimento de cálculo envolve apenas a multiplicação de duas matrizes para cada nível de transformação. Se $A[\cdot][\cdot]$ é a matriz de coeficientes dos filtros e $B[\cdot]$ é o sinal original, então $C[\cdot] = A[\cdot][\cdot]B[\cdot]$ corresponde ao sinal transformado, sendo que a disposição dos coeficientes nas matrizes é a seguinte:

$$A[\cdot][\cdot] = \begin{pmatrix} h_0 & h_1 & h_2 & \dots & \dots & \dots & h_{n-1} & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ g_0 & g_1 & g_2 & \dots & \dots & \dots & g_{n-1} & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & h_0 & h_1 & h_2 & \dots & \dots & h_{n-1} & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \dots & \dots & g_{n-1} & 0 & 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \vdots & & & & & & & & & & & & & \\ \vdots & & & & & & & & & & & & & \\ h_{n-1} & 0 & 0 & 0 & \dots & \dots & \dots & 0 & 0 & h_0 & h_1 & \dots & \dots & h_{n-3} & h_{n-2} \\ g_{n-1} & 0 & 0 & 0 & \dots & \dots & \dots & 0 & 0 & g_0 & g_1 & \dots & \dots & g_{n-3} & g_{n-2} \end{pmatrix},$$

$$B[\cdot] = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ \dots \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{pmatrix},$$

$$C[\cdot] = \begin{pmatrix} c_0 \\ c_{\frac{n}{2}} \\ c_1 \\ c_{\frac{n}{2}+1} \\ \dots \\ \vdots \\ c_{n-1} \\ c_{\frac{n}{2}-1} \end{pmatrix}.$$

Pode-se notar na matriz $A[\cdot][\cdot]$ que dois procedimentos estão embutidos no algoritmo de Mallat: *downsampling* e *wrap-around* [15]. O primeiro já foi mencionado anteriormente, e o segundo consiste em fazer com que os últimos coeficientes dos filtros ocupem as posições iniciais de cada linha. Isso faz com que a DWT tenha sempre o mesmo número de elementos do sinal original.

2.3.2 Cálculo da DWT inversa (IDWT)

Da mesma forma como ocorre no cálculo da DWT, para calcular a IDWT através do algoritmo de Mallat, apenas $\bar{h}[\cdot]$ e $\bar{g}[\cdot]$ são necessários. O cálculo procede de forma a obter novamente o vetor do sinal original $B[\cdot]$ a partir da multiplicação de $A^{-1}[\cdot][\cdot]$ por $C[\cdot]$, onde $A^{-1}[\cdot][\cdot]$, que é a inversa de $A[\cdot][\cdot]$, corresponde à matriz dos coeficientes dos filtros de síntese, isto é, $\bar{h}[\cdot]$ e $\bar{g}[\cdot]$. Tendo em vista que $A[\cdot][\cdot]$ é sempre ortogonal, por construção, $A^{-1}[\cdot][\cdot] = A^T[\cdot][\cdot]$, o que facilita muito a inversão da transformada

2.3.3 Momentos Nulos

A quantidade de momentos nulos [15] é uma propriedade interessante da DWT. Ela implica que, para um sinal que pode ser (aproximadamente) descrito por um polinômio de grau menor que M e uma *wavelet* que possui M momentos nulos, os coeficientes de detalhamento serão (aproximadamente) zero. Embora esse fato seja primordialmente importante nos esquemas de compressão de dados, ele pode ser levado em conta no presente trabalho por ter ligação com as características dos filtros. O m -ésimo momento pode ser calculado como $m = \sum_{k=0}^{p-1} t_k^m \psi(t_k)$, sendo p a quantidade de pontos da função *wavelet*, m o momento desejado e t cada ponto onde a função pode possuir valor diferente de 0 ($t = \frac{1}{2}s$, em que s é um escalar inteiro maior ou igual a 0).

2.3.4 Famílias de Transformadas *Wavelet*

Conforme ilustra a tabela 2.1 e as figuras 2.12, 2.13 e 2.14, as diversas famílias de filtros *wavelet* existentes [15] diferem no suporte dos filtros, assim como nas características de resposta em frequência e fase dos mesmos, o que faz com que

as funções ϕ e ψ também sofram reflexo de tais diferenças. Serão utilizadas neste trabalho as *wavelets* de Haar, Daubechies, Symmlets, Coiflets, Vaidyanathan e Beylkin, com diversos suportes, todas constituindo filtros com respostas ao impulso finitas (*Finite Impulse Response* - FIR), isto é, limitadas [3], sendo que as respostas em frequência se aproximam das ideais à medida que o suporte cresce.

Tabela 2.1: Características das famílias de *wavelets* utilizadas no presente trabalho, incluindo a quantidade de momentos da função *wavelet*.

Família	Suporte(n)	Fase	Observação	Momentos
Haar	2	linear	é a mais simples das <i>wavelets</i> , criada por Alfred Haar [20][15]	1
Daubechies	par, maior que 4	não linear	resposta ao impulso <i>maximally flat</i> , criada por Ingrid Daubechies [20][15]	$\frac{n}{2}$
Symmlets	par, múltiplo de 8	não linear	resposta ao impulso mais simétrica[20][15]	$\frac{n}{2} - 2$
Coiflets	par, múltiplo de 6	quase linear	resposta ao impulso quase simétrica, criada por Ronald Coifman [20][15]	$\frac{n}{2} - 1$
Vaidyanathan	24	não linear	otimizada para voz, criada por P. P. Vaidyanathan [20][15]	—
Beylkin	18	não linear	otimizada para áudio em geral [20][15]	$\frac{n}{2} - 2$

Figura 2.12: Formato das respostas ao impulso dos filtros *wavelet* de Haar, Daubechies, Vaidyanathan, Beylkin, Coiflet e Symmlet, respectivamente

Figura 2.13: Formatos das funções *scaling* dos filtros *wavelet* de Haar, Daubechies, Vaidyanathan, Beylkin, Coiflet e Symmlet, respectivamente.

Figura 2.14: Formatos das funções *wavelet* dos filtros *wavelet* de Haar, Daubechies, Vaidyanathan, Beylkin, Coiflet e Symmlet, respectivamente.

Capítulo 3

Descrição da Técnica Proposta

Este capítulo descreve o sistema proposto, incluindo a arquitetura do mesmo e cada passo do algoritmo associado, entre outros detalhes.

3.1 A Arquitetura do Sistema Proposto

Basicamente, o sistema proposto para segmentação de voz é constituído por um módulo de segmentação do sinal de entrada, um módulo de análise fractal e, por fim, um módulo de decisão composto por duas etapas de teste, conforme ilustra a figura 3.2 ao fim do capítulo.

O módulo de segmentação recebe o sinal de voz e separa-o em *frames* de comprimento s cada um, onde s deve ser uma potência de 2. Empiricamente, os testes realizados têm demonstrado que o melhor valor para s é 256. Se $s = 128$ ou $s = 512$ forem utilizados, os resultados pioram, possivelmente devido ao fato de que *frames* curtos não capturam ao menos um período de determinados sinais periódicos e *frames* longos podem englobar mais de uma transição que não seria,

nesse caso, apontada pelo algoritmo.

O módulo de análise fractal, constituído por uma única etapa, tem a função de obter a dimensão do fractal, D_i , de cada um dos *frames* i . Já o módulo de decisão é composto por duas etapas de teste, A e B . A primeira é responsável por verificar a variação entre as dimensões dos fractais de dois *frames* consecutivos, enquanto que a segunda possui a função de associar essa variação, caso seja significativa, com a energia do trecho correspondente do sinal original, decidindo se o evento corresponde à uma transição entre palavras ou fonemas.

A descrição completa e detalhada do algoritmo, encontra-se a seguir.

3.2 O Algoritmo Proposto

A descrição detalhada das etapas do algoritmo proposto segue.

- **INÍCIO**
- **PASSO 1:** Segmente o sinal de voz de entrada, $f[\cdot]$, de comprimento C , em *frames* com $s = 256$ amostras cada, descartando as amostras finais do sinal, caso não sejam suficientes para formar um frame;
- **PASSO 2:** Para cada *frame* $r_i[\cdot]$, ($0 \leq i \leq n - 1$), repita:
 - **INÍCIO LAÇO:**
 - **PASSO 2.1:** Normalize $r_i[\cdot]$ de forma que apresente média 0, subtraindo a média de todas as amostras, e também energia unitária;

- **PASSO 2.2:** Obtenha a árvore de decomposição *wavelet* nível j de r usando a família de filtros w , sendo $j = \frac{\log(s)}{\log(2)}$ e w especificado à frente;
- **PASSO 2.3:** Considerando somente os detalhes de cada nível de decomposição, descartando as aproximações, preencha os vetores $bl[\cdot]$ e $be[\cdot]$, ambos de tamanho j , da seguinte forma:

$$bl_i = comprimento_da_sub_banda_i$$

$$be_i = \frac{1}{bl_i} \sum_{k \in sub_banda_i} DWT(r)_k^2, \quad ,$$

sendo $(0 \leq i \leq j - 1)$. Dessa forma, a dimensão do fractal correspondente, D_i , $(1 \leq D_i \leq 2)$, é

$$D_i = 2 - |H_i|, \quad ,$$

onde $H_i = \frac{\beta-1}{2}$ corresponde ao expoente de Hurst, de acordo com [11], e

$$\beta = \frac{j \sum_{k=0}^{j-1} \log_2(bl_k) \log_2(be_k) - \sum_{k=0}^{j-1} \log_2(be_k) \sum_{k=0}^{j-1} \log_2(bl_k)}{j \sum_{k=0}^{j-1} \log_2(bl_k)^2 - \left(\sum_{k=0}^{j-1} \log_2(bl_k) \right)^2};$$

- **PASSO 2.4:** Preencher a próxima posição livre do vetor $fd[\cdot]$, que armazena o valor de D_i , referente ao *frame* corrente r ;
- **FIM LAÇO;**
- **PASSO 3:** Obter a derivada, seguida do “envelope”, do sinal contido no vetor $fd[\cdot]$, por meio de uma filtragem nesse sinal, conforme explicado adiante;

- **PASSO 4:** Identificar as bordas de subida e descida no envelope, considerando os pontos de pico entre elas como sendo transições. Picos associados com regiões de baixa energia na parte correspondente do sinal original indicam transições entre palavras, e picos associados com regiões de alta energia no sinal original indicam transições entre fonemas;
- **FIM.**

No passo 1 do algoritmo, a última parte do sinal é descartada no caso de seu comprimento ser menor do que 256 amostras. Esse fato não traz problemas tendo em vista que trechos com comprimento menor do que 256 não contêm fonemas significativos, do ponto de vista auditivo, após uma possível transição, conforme pode-se observar na prática.

No passo 2.1, a normalização realizada para que o sinal possua média 0 consiste em retirar o componente contínuo do sinal, caso exista, sendo implementada da forma

$$f_i \leftarrow f_i - \frac{\left(\sum_{k=0}^C f_k \right)}{C} .$$

Já a normalização aplicada para fins de tornar a energia do sinal unitária, que tem por objetivo delinear a correta especificação do limiar de energia do passo 4, é da forma

$$f_i \leftarrow \frac{f_i}{\sqrt{\left(\sum_{k=0}^C f_k^2 \right)}} .$$

No passo 2.2, a árvore de decomposição *wavelet* é obtida com o uso do algoritmo de Mallat, conforme a revisão de literatura apresentada. A família de filtros w que deve ser utilizada nesse passo deve ser escolhida com base em algumas

considerações. De acordo com os conceitos sobre DWT revisados no capítulo 2, reforçados pelas ilustrações contidas na figura 2.12, pode-se observar que filtragens passa-baixas e passa-altas ocorrem durante a análise de um sinal de voz $f[\cdot]$ e portanto, o comprimento dos filtros, N , é responsável pela seletividade de frequências, Q , e também resolução temporal, R . Se Q for pobre, as diferentes sub-bandas de decomposição estarão contaminadas com as frequências das sub-bandas adjacentes, produzindo possivelmente erro no cálculo da dimensão do fractal. Se por outro lado, R for pobre, a localização temporal das transições entre fonemas ficará comprometida. Infelizmente, não é possível atender a ambas as restrições para otimizar Q e R : o aumento de N aumenta R , mas diminui Q e, a redução de N aumenta Q , mas diminui R . Uma outra característica interessante é a fase linear dos filtros utilizados que, sem dúvida, é desejada para evitar atrasos diferentes em cada sub-banda de frequências. Dessa forma, filtros com resposta simétrica ao impulso ou anti-simétrica são preferíveis [3].

Visando atender as especificações apresentadas anteriormente, N deve ser balanceado para equilibrar Q e R . De acordo com resultados anteriores de pesquisas desenvolvidas pelo *SpeechLab* / IFSC / USP, tais como a documentada em [1], N entre 12 e 24 é sugerido, devido ao fato de que, fora dessa faixa, Q ou R será certamente não ótimo. Com relação à questão da fase dos filtros, o filtro de Haar é o único que atende a característica de resposta ao impulso simétrica, entretanto, seu suporte é $N = 2$ e, portanto, ele não satisfaz a restrição que visa equilibrar Q e R . Assim, a única possibilidade é utilizar filtros *wavelet* que possuam respostas ao impulso quase simétrica. A figura 2.12 do capítulo 2 sugere, portanto, o uso da família Symmlets, sendo $N = 16$ o suporte indicado. Apesar dessas considerações

teóricas para a escolha de w , testes foram realizados com todas as famílias apresentadas na revisão de literatura, usando diversos suportes. Comentários adicionais relatando tais experiências estão inclusos no próximo capítulo.

No passo 2.3 do algoritmo, os vetores $bl[\cdot]$ e $be[\cdot]$ armazenam, respectivamente, o comprimento de cada banda da decomposição e a energia das amostras nelas contida. Os valores presentes em cada um desses vetores são utilizados para obter o valor de β , que corresponde ao coeficiente angular da reta média que interpola os logaritmos, na base 2, de tais pontos, obtida, por sua vez, pelo ajuste de regressão linear via Método dos Mínimos Quadrados [21]. A partir de β , o expoente de Hurst é tal que $H = \frac{\beta-1}{2}$ e a dimensão do fractal correspondente é tal que $D = 2 - |H|$. Os valores de D para cada um dos *frames* de voz são armazenados no vetor $fd[\cdot]$ como descreve o passo 2.4.

No passo 3, a derivada obtida é igual a diferencial do sinal, ou seja, a diferença ponto a ponto, em módulo. Uma filtragem via convolução com um filtro de resposta ao impulso finita, de ordem 20 e com corte em um vigésimo da máxima frequência, é aplicada para tornar o sinal presente em $fd[\cdot]$ mais suave, obtendo assim o “envelope” dessa curva. Um exemplo de um sinal contido em $fd[\cdot]$ e seu respectivo envelope encontram-se na figura 3.1.

O passo 4, que corresponde as etapas de teste do módulo de decisão, consiste basicamente em dois testes. O primeiro é responsável por identificar os picos no envelope obtido no passo 3. Tais picos correspondem às variações mais consideráveis entre os valores das dimensões dos fractais de *frames* consecutivos.

Figura 3.1: (acima): exemplo de um sinal contido no vetor $fd[\cdot]$; (abaixo): respectivo envelope.

Tais pontos apontam, de acordo com o critério defendido no presente trabalho, a presença de transições, caracterizadas por mudanças no comportamento estatístico das amostras contidas nos *frames* em questão. Essas alterações de comportamento do envelope podem refletir transições de fonemas para silêncio, ou vice-versa, caracterizando portanto transições entre palavras, ou podem refletir transições entre fonemas consecutivos de uma mesma palavra. Tal decisão se dá com base na energia normalizada do sinal original correspondente, isto é, uma transição associada com um valor de energia abaixo de um certo limiar, que aponta a presença de silêncio, indica transição entre palavras; já uma transição associada com um valor de energia acima desse limiar sugere uma transição entre fonemas de uma mesma palavra [4]. Essa última análise caracteriza a segunda etapa de teste.

Todo o algoritmo foi implementado em linguagem C++ sob ambiente LINUX [22], sendo que o código-fonte correspondente encontra-se no apêndice II. O próximo capítulo descreve os testes e resultados obtidos com o algoritmo proposto.

Figura 3.2: Arquitetura básica do sistema proposto. Legenda: N.C.T.: não contém transição de fonema ou palavra ; V.N.S.: variação não significativa entre as dimensões dos fractais dos *frames* consecutivos sob análise ; V.S.: variação significativa entre as dimensões dos fractais dos *frames* consecutivos sob análise ; T.P.: transição entre palavras ; T.F.: transição entre fonemas ; A.E.S.: alta energia no trecho correspondente do sinal original ; B.E.S.: baixa energia no trecho correspondente do sinal original.

Capítulo 4

Testes e Resultados

Neste capítulo encontram-se os testes realizados, que foram divididos em cinco grupos, assim como os respectivos resultados obtidos. Diversos filtros wavelet foram experimentados, variando a família e o suporte dos mesmos para obtenção das dimensões dos fractais e para fins de realizar as segmentações de palavras e fonemas.

4.1 Materiais e Métodos

Todos os sinais analisados correspondem às sentenças extraídas do diretório *test* da base TIMIT do LDC, conforme mencionado anteriormente, tendo sido amostrados a uma taxa de 16000 amostras por segundo, 16 bits - *wave*. Para cada sentença, além do arquivo *wave* que contém o sinal, três outros arquivos em formato texto estão disponíveis. O primeiro, com a extensão *.txt*, corresponde à sentença propriamente dita. O segundo, com a extensão *.wrđ*, aponta em quais

amostras ocorrem as transições entre cada palavra. Por fim, o terceiro arquivo, com a extensão *.phn* aponta as correspondentes transições entre os fonemas. Tais pontos de transições entre palavras e fonemas levam em conta uma segmentação realizada por inspeção, sendo considerada plenamente aceitável pela comunidade internacional na área de processamento de voz. Dessa forma, os testes realizados a seguir apresentam uma comparação entre esses pontos de transição e os apontados pelo algoritmo proposto e, ainda, um teste de audição para confirmar que, de fato, as palavras ou fonemas respectivos foram ouvidos.

4.2 Bateria de Testes

A bateria completa dos testes realizados segue, incluindo diversas tabelas, assim como os resultados obtidos, cuja discussão e interpretação, que levam às conclusões finais, encontram-se no próximo capítulo.

Os testes estão divididos em cinco grupos. O primeiro deles, que apresenta os resultados inclusos nas tabelas de 4.1 até 4.11, tem a função de verificar apenas a precisão dos valores obtidos nos cálculos das dimensões (D) dos fractais de alguns sinais sintetizados, sem nenhuma relação com qualquer processo de segmentação de voz. Os sinais em questão foram sintetizados por método computacional para alguns valores na faixa de 1.0 até 2.0. As comparações envolvem os métodos *box-counting*, *walking-divider*, *prism*, *power spectrum* baseado na Transformada de Fourier e, por fim, o método proposto com o uso de diversos filtros *wavelet* de diversos suportes.

O segundo grupo de testes concentra-se nas transições entre palavras, comparando

para diversas sentenças da TIMIT, os pontos indicados como transições na base de dados com os respectivos pontos apresentados pelo algoritmo proposto. Esses testes estão demonstrados nas tabelas 4.12 até 4.16. Já o terceiro grupo de testes, exibido nas tabelas 4.17 até 4.21, realiza o mesmo trabalho, mas considerando fonemas ao invés de palavras. Por fim, os dois últimos grupos de testes utilizam uma sentença particular da TIMIT para exemplificar como a variação dos filtros *wavelet* no algoritmo proposto interfere nas segmentações de palavras e fonemas, conforme apresentado nas tabelas 4.22 e 4.23, e 4.24 e 4.25, respectivamente.

A partir das segmentações produzidas pelo algoritmo proposto, testes auditivos também foram realizados para confirmar cada segmentação. É importante destacar que as sentenças utilizadas são de conteúdo fonético rico, contendo trechos vozeados e não-vozeados, assegurando a experimentação para variados casos, cobrindo, portanto, uma ampla gama de características.

O próximo capítulo apresenta a discussão detalhada dos resultados apresentados nas tabelas seguintes.

Tabela 4.1: Testes do **grupo 1** para os diversos métodos apresentados para um fractal sintetizado, sendo que as implementações estão baseadas em [11] e na técnica proposta. [Resultados para $D = 1.0$]. Legenda: MET: método utilizado ; SW: suporte da *wavelet*, quando for o caso; VO: valor obtido para a dimensão do fractal. Quando MET for o nome de uma *wavelet*, o método corresponde à técnica proposta utilizando a referida *wavelet*.

MET	SW	VO	MET	SW	VO	MET	SW	VO	MET	SW	VO
Haar	2	1.0	Daubechies	4	1.0	Daubechies	6	1.0	Daubechies	8	1.0
Daubechies	10	1.0	Daubechies	12	1.0	Daubechies	14	1.0	Daubechies	16	1.0
Daubechies	18	1.0	Daubechies	20	1.0	Daubechies	22	1.0	Daubechies	24	1.0
Daubechies	26	1.0	Daubechies	28	1.0	Daubechies	30	1.0	Daubechies	32	1.0
Daubechies	34	1.0	Daubechies	36	1.0	Daubechies	38	1.0	Daubechies	40	1.0
Daubechies	42	1.0	Daubechies	44	1.0	Daubechies	46	1.0	Daubechies	48	1.0
Daubechies	50	1.0	Daubechies	52	1.0	Daubechies	54	1.0	Daubechies	56	1.0
Daubechies	58	1.0	Daubechies	60	1.0	Daubechies	62	1.0	Daubechies	64	1.0
Daubechies	66	1.0	Daubechies	68	1.0	Daubechies	70	1.0	Daubechies	72	1.0
Daubechies	74	1.0	Daubechies	76	1.0	Coiflet	6	1.0	Coiflet	12	1.0
Coiflet	18	1.0	Coiflet	24	1.0	Coiflet	30	1.0	Symmet	8	1.0
Symmet	16	1.0	Beylkin	18	1.0	Vaidyanathan	24	1.0	box-counting	-	1.097
walk-divider	-	1.220	prism	-	1.110	power-spectrum	-	1.006			

Tabela 4.2: Testes do **grupo 1** para os diversos métodos apresentados para um fractal sintetizado, sendo que as implementações estão baseadas em [11] e na técnica proposta. [Resultados para $D = 1.1$]. Legenda: MET: método utilizado ; SW: suporte da *wavelet*, quando for o caso; VO: valor obtido para a dimensão do fractal. Quando MET for o nome de uma *wavelet*, o método corresponde à técnica proposta utilizando a referida *wavelet*.

MET	SW	VO	MET	SW	VO	MET	SW	VO	MET	SW	VO
Haar	2	1.151	Daubechies	4	1.149	Daubechies	6	1.147	Daubechies	8	1.146
Daubechies	10	1.146	Daubechies	12	1.146	Daubechies	14	1.145	Daubechies	16	1.142
Daubechies	18	1.138	Daubechies	20	1.137	Daubechies	22	1.137	Daubechies	24	1.136
Daubechies	26	1.131	Daubechies	28	1.132	Daubechies	30	1.134	Daubechies	32	1.132
Daubechies	34	1.128	Daubechies	36	1.128	Daubechies	38	1.123	Daubechies	40	1.126
Daubechies	42	1.123	Daubechies	44	1.121	Daubechies	46	1.121	Daubechies	48	1.121
Daubechies	50	1.122	Daubechies	52	1.125	Daubechies	54	1.121	Daubechies	56	1.120
Daubechies	58	1.119	Daubechies	60	1.123	Daubechies	62	1.122	Daubechies	64	1.121
Daubechies	66	1.171	Daubechies	68	1.162	Daubechies	70	1.182	Daubechies	72	1.181
Daubechies	74	1.180	Daubechies	76	1.181	Coiflet	6	1.142	Coiflet	12	1.142
Coiflet	18	1.161	Coiflet	24	1.142	Coiflet	30	1.071	Symmet	8	1.131
Symmet	16	1.120	Beylkin	18	1.121	Vaidyanathan	24	1.142	box-counting	-	1.147
walk-divider	-	1.259	prism	-	1.160	power-spectrum	-	1.138			

Tabela 4.3: Testes do **grupo 1** para os diversos métodos apresentados para um fractal sintetizado, sendo que as implementações estão baseadas em [11] e na técnica proposta. [Resultados para $D = 1.2$]. Legenda: MET: método utilizado ; SW: suporte da *wavelet*, quando for o caso; VO: valor obtido para a dimensão do fractal. Quando MET for o nome de uma *wavelet*, o método corresponde à técnica proposta utilizando a referida *wavelet*.

MET	SW	VO	MET	SW	VO	MET	SW	VO	MET	SW	VO
Haar	2	1.311	Daubechies	4	1.298	Daubechies	6	1.287	Daubechies	8	1.276
Daubechies	10	1.276	Daubechies	12	1.269	Daubechies	14	1.275	Daubechies	16	1.265
Daubechies	18	1.264	Daubechies	20	1.263	Daubechies	22	1.262	Daubechies	24	1.264
Daubechies	26	1.254	Daubechies	28	1.256	Daubechies	30	1.258	Daubechies	32	1.252
Daubechies	34	1.252	Daubechies	36	1.244	Daubechies	38	1.241	Daubechies	40	1.237
Daubechies	42	1.232	Daubechies	44	1.231	Daubechies	46	1.231	Daubechies	48	1.229
Daubechies	50	1.225	Daubechies	52	1.228	Daubechies	54	1.227	Daubechies	56	1.227
Daubechies	58	1.238	Daubechies	60	1.240	Daubechies	62	1.236	Daubechies	64	1.236
Daubechies	66	1.239	Daubechies	68	1.237	Daubechies	70	1.239	Daubechies	72	1.239
Daubechies	74	1.239	Daubechies	76	1.243	Coiflet	6	1.227	Coiflet	12	1.225
Coiflet	18	1.274	Coiflet	24	1.228	Coiflet	30	1.229	Symmet	8	1.224
Symmet	16	1.221	Beylkin	18	1.224	Vaidyanathan	24	1.223	box-counting	-	1.165
walk-divider	-	1.294	prism	-	1.301	power-spectrum	-	1.219			

Tabela 4.4: Testes do **grupo 1** para os diversos métodos apresentados para um fractal sintetizado, sendo que as implementações estão baseadas em [11] e na técnica proposta. [Resultados para $D = 1.3$]. Legenda: MET: método utilizado ; SW: suporte da *wavelet*, quando for o caso; VO: valor obtido para a dimensão do fractal. Quando MET for o nome de uma *wavelet*, o método corresponde à técnica proposta utilizando a referida *wavelet*.

MET	SW	VO	MET	SW	VO	MET	SW	VO	MET	SW	VO
Haar	2	1.387	Daubechies	4	1.288	Daubechies	6	1.278	Daubechies	8	1.287
Daubechies	10	1.312	Daubechies	12	1.315	Daubechies	14	1.314	Daubechies	16	1.318
Daubechies	18	1.321	Daubechies	20	1.322	Daubechies	22	1.322	Daubechies	24	1.321
Daubechies	26	1.327	Daubechies	28	1.326	Daubechies	30	1.332	Daubechies	32	1.331
Daubechies	34	1.338	Daubechies	36	1.337	Daubechies	38	1.341	Daubechies	40	1.346
Daubechies	42	1.355	Daubechies	44	1.349	Daubechies	46	1.335	Daubechies	48	1.332
Daubechies	50	1.315	Daubechies	52	1.321	Daubechies	54	1.316	Daubechies	56	1.319
Daubechies	58	1.315	Daubechies	60	1.317	Daubechies	62	1.320	Daubechies	64	1.321
Daubechies	66	1.319	Daubechies	68	1.318	Daubechies	70	1.320	Daubechies	72	1.322
Daubechies	74	1.321	Daubechies	76	1.321	Coiflet	6	1.321	Coiflet	12	1.317
Coiflet	18	1.319	Coiflet	24	1.316	Coiflet	30	1.314	Symmlet	8	1.319
Symmlet	16	1.311	Beylkin	18	1.314	Vaidyanathan	24	1.314	box-counting	-	1.261
walk-divider	-	1.351	prism	-	1.348	power-spectrum	-	1.273			

Tabela 4.5: Testes do **grupo 1** para os diversos métodos apresentados para um fractal sintetizado, sendo que as implementações estão baseadas em [11] e na técnica proposta. [Resultados para $D = 1.4$]. Legenda: MET: método utilizado ; SW: suporte da *wavelet*, quando for o caso; VO: valor obtido para a dimensão do fractal. Quando MET for o nome de uma *wavelet*, o método corresponde à técnica proposta utilizando a referida *wavelet*.

MET	SW	VO	MET	SW	VO	MET	SW	VO	MET	SW	VO
Haar	2	1.377	Daubechies	4	1.379	Daubechies	6	1.391	Daubechies	8	1.411
Daubechies	10	1.419	Daubechies	12	1.423	Daubechies	14	1.422	Daubechies	16	1.427
Daubechies	18	1.422	Daubechies	20	1.430	Daubechies	22	1.428	Daubechies	24	1.433
Daubechies	26	1.432	Daubechies	28	1.432	Daubechies	30	1.430	Daubechies	32	1.429
Daubechies	34	1.433	Daubechies	36	1.435	Daubechies	38	1.431	Daubechies	40	1.432
Daubechies	42	1.432	Daubechies	44	1.433	Daubechies	46	1.432	Daubechies	48	1.432
Daubechies	50	1.431	Daubechies	52	1.437	Daubechies	54	1.433	Daubechies	56	1.436
Daubechies	58	1.432	Daubechies	60	1.430	Daubechies	62	1.428	Daubechies	64	1.428
Daubechies	66	1.426	Daubechies	68	1.423	Daubechies	70	1.422	Daubechies	72	1.419
Daubechies	74	1.419	Daubechies	76	1.420	Coiflet	6	1.431	Coiflet	12	1.432
Coiflet	18	1.439	Coiflet	24	1.444	Coiflet	30	1.423	Symmet	8	1.413
Symmet	16	1.412	Beylkin	18	1.413	Vaidyanathan	24	1.411	box-counting	-	1.307
walk-divider	-	1.404	prism	-	1.391	power-spectrum	-	1.382			

Tabela 4.6: Testes do **grupo 1** para os diversos métodos apresentados para um fractal sintetizado, sendo que as implementações estão baseadas em [11] e na técnica proposta. [Resultados para $D = 1.5$]. Legenda: MET: método utilizado ; SW: suporte da *wavelet*, quando for o caso; VO: valor obtido para a dimensão do fractal. Quando MET for o nome de uma *wavelet*, o método corresponde à técnica proposta utilizando a referida *wavelet*.

MET	SW	VO	MET	SW	VO	MET	SW	VO	MET	SW	VO
Haar	2	1.491	Daubechies	4	1.494	Daubechies	6	1.492	Daubechies	8	1.507
Daubechies	10	1.509	Daubechies	12	1.511	Daubechies	14	1.513	Daubechies	16	1.515
Daubechies	18	1.515	Daubechies	20	1.517	Daubechies	22	1.515	Daubechies	24	1.523
Daubechies	26	1.521	Daubechies	28	1.521	Daubechies	30	1.516	Daubechies	32	1.523
Daubechies	34	1.523	Daubechies	36	1.522	Daubechies	38	1.522	Daubechies	40	1.526
Daubechies	42	1.528	Daubechies	44	1.526	Daubechies	46	1.529	Daubechies	48	1.531
Daubechies	50	1.522	Daubechies	52	1.523	Daubechies	54	1.524	Daubechies	56	1.523
Daubechies	58	1.521	Daubechies	60	1.523	Daubechies	62	1.521	Daubechies	64	1.525
Daubechies	66	1.521	Daubechies	68	1.522	Daubechies	70	1.524	Daubechies	72	1.524
Daubechies	74	1.522	Daubechies	76	1.519	Coiflet	6	1.516	Coiflet	12	1.512
Coiflet	18	1.511	Coiflet	24	1.506	Coiflet	30	1.504	Symmet	8	1.497
Symmet	16	1.498	Beylkin	18	1.516	Vaidyanathan	24	1.513	box-counting	-	1.380
walk-divider	-	1.451	prism	-	1.468	power-spectrum	-	1.495			

Tabela 4.7: Testes do **grupo 1** para os diversos métodos apresentados para um fractal sintetizado, sendo que as implementações estão baseadas em [11] e na técnica proposta. [Resultados para $D = 1.6$]. Legenda: MET: método utilizado ; SW: suporte da *wavelet*, quando for o caso; VO: valor obtido para a dimensão do fractal. Quando MET for o nome de uma *wavelet*, o método corresponde à técnica proposta utilizando a referida *wavelet*.

MET	SW	VO	MET	SW	VO	MET	SW	VO	MET	SW	VO
Haar	2	1.591	Daubechies	4	1.593	Daubechies	6	1.609	Daubechies	8	1.611
Daubechies	10	1.612	Daubechies	12	1.614	Daubechies	14	1.612	Daubechies	16	1.612
Daubechies	18	1.614	Daubechies	20	1.615	Daubechies	22	1.616	Daubechies	24	1.615
Daubechies	26	1.618	Daubechies	28	1.620	Daubechies	30	1.613	Daubechies	32	1.616
Daubechies	34	1.617	Daubechies	36	1.618	Daubechies	38	1.612	Daubechies	40	1.617
Daubechies	42	1.617	Daubechies	44	1.617	Daubechies	46	1.618	Daubechies	48	1.616
Daubechies	50	1.624	Daubechies	52	1.622	Daubechies	54	1.618	Daubechies	56	1.619
Daubechies	58	1.622	Daubechies	60	1.621	Daubechies	62	1.620	Daubechies	64	1.621
Daubechies	66	1.621	Daubechies	68	1.619	Daubechies	70	1.615	Daubechies	72	1.614
Daubechies	74	1.616	Daubechies	76	1.616	Coiflet	6	1.612	Coiflet	12	1.614
Coiflet	18	1.614	Coiflet	24	1.611	Coiflet	30	1.613	Symmlet	8	1.611
Symmlet	16	1.609	Beylkin	18	1.615	Vaidyanathan	24	1.614	box-counting	-	1.418
walk-divider	-	1.501	prism	-	1.523	power-spectrum	-	1.599			

Tabela 4.8: Testes do **grupo 1** para os diversos métodos apresentados para um fractal sintetizado, sendo que as implementações estão baseadas em [11] e na técnica proposta. [Resultados para $D = 1.7$]. Legenda: MET: método utilizado ; SW: suporte da *wavelet*, quando for o caso; VO: valor obtido para a dimensão do fractal. Quando MET for o nome de uma *wavelet*, o método corresponde à técnica proposta utilizando a referida *wavelet*.

<i>MET</i>	<i>SW</i>	<i>VO</i>	<i>MET</i>	<i>SW</i>	<i>VO</i>	<i>MET</i>	<i>SW</i>	<i>VO</i>	<i>MET</i>	<i>SW</i>	<i>VO</i>
Haar	2	1.689	Daubechies	4	1.719	Daubechies	6	1.721	Daubechies	8	1.719
Daubechies	10	1.719	Daubechies	12	1.718	Daubechies	14	1.721	Daubechies	16	1.723
Daubechies	18	1.718	Daubechies	20	1.719	Daubechies	22	1.723	Daubechies	24	1.722
Daubechies	26	1.731	Daubechies	28	1.727	Daubechies	30	1.722	Daubechies	32	1.729
Daubechies	34	1.738	Daubechies	36	1.737	Daubechies	38	1.735	Daubechies	40	1.734
Daubechies	42	1.731	Daubechies	44	1.734	Daubechies	46	1.736	Daubechies	48	1.733
Daubechies	50	1.739	Daubechies	52	1.737	Daubechies	54	1.733	Daubechies	56	1.735
Daubechies	58	1.729	Daubechies	60	1.728	Daubechies	62	1.733	Daubechies	64	1.732
Daubechies	66	1.736	Daubechies	68	1.735	Daubechies	70	1.736	Daubechies	72	1.739
Daubechies	74	1.741	Daubechies	76	1.741	Coiflet	6	1.723	Coiflet	12	1.722
Coiflet	18	1.721	Coiflet	24	1.721	Coiflet	30	1.719	Symmet	8	1.716
Symmet	16	1.706	Beylkin	18	1.713	Vaidyanathan	24	1.713	<i>box-counting</i>	-	1.482
<i>walk-divider</i>	-	1.542	<i>prism</i>	-	1.556	<i>power-spectrum</i>	-	1.705			

Tabela 4.9: Testes do **grupo 1** para os diversos métodos apresentados para um fractal sintetizado, sendo que as implementações estão baseadas em [11] e na técnica proposta. [Resultados para $D = 1.8$]. Legenda: MET: método utilizado ; SW: suporte da *wavelet*, quando for o caso; VO: valor obtido para a dimensão do fractal. Quando MET for o nome de uma *wavelet*, o método corresponde à técnica proposta utilizando a referida *wavelet*.

MET	SW	VO	MET	SW	VO	MET	SW	VO	MET	SW	VO
Haar	2	1.791	Daubechies	4	1.793	Daubechies	6	1.812	Daubechies	8	1.814
Daubechies	10	1.814	Daubechies	12	1.818	Daubechies	14	1.814	Daubechies	16	1.817
Daubechies	18	1.819	Daubechies	20	1.823	Daubechies	22	1.823	Daubechies	24	1.819
Daubechies	26	1.821	Daubechies	28	1.821	Daubechies	30	1.823	Daubechies	32	1.826
Daubechies	34	1.822	Daubechies	36	1.828	Daubechies	38	1.823	Daubechies	40	1.823
Daubechies	42	1.831	Daubechies	44	1.834	Daubechies	46	1.824	Daubechies	48	1.831
Daubechies	50	1.821	Daubechies	52	1.828	Daubechies	54	1.829	Daubechies	56	1.823
Daubechies	58	1.823	Daubechies	60	1.829	Daubechies	62	1.823	Daubechies	64	1.824
Daubechies	66	1.827	Daubechies	68	1.828	Daubechies	70	1.822	Daubechies	72	1.823
Daubechies	74	1.829	Daubechies	76	1.822	Coiflet	6	1.813	Coiflet	12	1.813
Coiflet	18	1.812	Coiflet	24	1.814	Coiflet	30	1.815	Symmlet	8	1.812
Symmlet	16	1.811	Beylkin	18	1.821	Vaidyanathan	24	1.819	box-counting	-	1.537
walk-divider	-	1.592	prism	-	1.623	power-spectrum	-	1.832			

Tabela 4.10: Testes do **grupo 1** para os diversos métodos apresentados para um fractal sintetizado, sendo que as implementações estão baseadas em [11] e na técnica proposta. [Resultados para $D = 1.9$]. Legenda: MET: método utilizado ; SW: suporte da *wavelet*, quando for o caso; VO: valor obtido para a dimensão do fractal. Quando MET for o nome de uma *wavelet*, o método corresponde à técnica proposta utilizando a referida *wavelet*.

MET	SW	VO	MET	SW	VO	MET	SW	VO	MET	SW	VO
Haar	2	1.878	Daubechies	4	1.886	Daubechies	6	1.909	Daubechies	8	1.913
Daubechies	10	1.913	Daubechies	12	1.916	Daubechies	14	1.919	Daubechies	16	1.923
Daubechies	18	1.921	Daubechies	20	1.923	Daubechies	22	1.930	Daubechies	24	1.929
Daubechies	26	1.923	Daubechies	28	1.929	Daubechies	30	1.928	Daubechies	32	1.923
Daubechies	34	1.921	Daubechies	36	1.931	Daubechies	38	1.921	Daubechies	40	1.929
Daubechies	42	1.931	Daubechies	44	1.928	Daubechies	46	1.928	Daubechies	48	1.925
Daubechies	50	1.925	Daubechies	52	1.925	Daubechies	54	1.927	Daubechies	56	1.925
Daubechies	58	1.924	Daubechies	60	1.923	Daubechies	62	1.928	Daubechies	64	1.924
Daubechies	66	1.922	Daubechies	68	1.925	Daubechies	70	1.925	Daubechies	72	1.924
Daubechies	74	1.920	Daubechies	76	1.921	Coiflet	6	1.914	Coiflet	12	1.911
Coiflet	18	1.913	Coiflet	24	1.911	Coiflet	30	1.909	Symmlet	8	1.906
Symmlet	16	1.897	Beylkin	18	1.911	Vaidyanathan	24	1.908	box-counting	-	1.554
walk-divider	-	1.617	prism	-	1.702	power-spectrum	-	1.942			

Tabela 4.11: Testes do **grupo 1** para os diversos métodos apresentados para um fractal sintetizado, sendo que as implementações estão baseadas em [11] e na técnica proposta. [Resultados para $D = 2.0$]. Legenda: MET: método utilizado ; SW: suporte da *wavelet*, quando for o caso; VO: valor obtido para a dimensão do fractal. Quando MET for o nome de uma *wavelet*, o método corresponde à técnica proposta utilizando a referida *wavelet*.

MET	SW	VO	MET	SW	VO	MET	SW	VO	MET	SW	VO
Haar	2	1.921	Daubechies	4	1.961	Daubechies	6	1.989	Daubechies	8	1.993
Daubechies	10	1.994	Daubechies	12	1.995	Daubechies	14	1.994	Daubechies	16	1.998
Daubechies	18	1.994	Daubechies	20	1.992	Daubechies	22	1.994	Daubechies	24	1.996
Daubechies	26	1.997	Daubechies	28	1.996	Daubechies	30	1.997	Daubechies	32	1.996
Daubechies	34	1.993	Daubechies	36	1.993	Daubechies	38	1.997	Daubechies	40	1.997
Daubechies	42	1.998	Daubechies	44	1.996	Daubechies	46	1.997	Daubechies	48	1.993
Daubechies	50	1.995	Daubechies	52	1.996	Daubechies	54	1.997	Daubechies	56	1.996
Daubechies	58	1.997	Daubechies	60	1.997	Daubechies	62	1.996	Daubechies	64	1.994
Daubechies	66	1.989	Daubechies	68	1.992	Daubechies	70	1.991	Daubechies	72	1.991
Daubechies	74	1.994	Daubechies	76	1.994	Coiflet	6	1.992	Coiflet	12	1.991
Coiflet	18	1.990	Coiflet	24	1.992	Coiflet	30	1.981	Symmet	8	1.991
Symmet	16	1.998	Beylkin	18	1.996	Vaidyanathan	24	1.990	box-counting	-	1.561
walk-divider	-	1.604	prism	-	1.688	power-spectrum	-	1.997			

Tabela 4.12: Testes do **grupo 2**: sentença e transições entre as palavras, de acordo com a documentação da base TIMIT (TR TIMIT) e de acordo com os testes obtidos com o método proposto (TR MP).

sentença	TR TIMIT	TR MP
<i>She had your dark suit in greasy wash water all year</i>	12783	12800
	17103	17152
	18760	21504
	24104	24064
	29179	29184
	31880	31744
	38568	38400
	45624	45312
	52378	52480
	55461	55552
	60600	60416

Tabela 4.13: Testes do **grupo 2**: sentença e transições entre as palavras, de acordo com a documentação da base TIMIT (TR TIMIT) e de acordo com os testes obtidos com o método proposto (TR MP).

sentença	TR TIMIT	TR MP
<i>Dont ask me to carry an oily rag like that</i>	15907	15872
	19990	19968
	23120	23040
	24680	24576
	31248	30976
	33149	33024
	39080	38912
	44850	44800
	49040	49152
	55000	55296

Tabela 4.14: Testes do **grupo 2**: sentença e transições entre as palavras, de acordo com a documentação da base TIMIT (TR TIMIT) e de acordo com os testes obtidos com o método proposto (TR MP).

sentença	TR TIMIT	TR MP
<i>His captain was thin and haggard and his beautiful boots were worn and shabby</i>	20161	21760
	24187	24320
	27710	27648
	30036	30696
	38210	38144
	40480	40192
	43550	43520
	51080	51456
	58720	58880
	60859	60928
	66800	66560
	68560	68608
	69000	69120
	76867	76800

Tabela 4.15: Testes do **grupo 2**: sentença e transições entre as palavras, de acordo com a documentação da base TIMIT (TR TIMIT) e de acordo com os testes obtidos com o método proposto (TR MP).

sentença	TR TIMIT	TR MP
<i>The reasons for this dive seemed foolish now</i>	21040	21248
	23227	23296
	27560	27648
	33480	33536
	40370	39936
	49925	50176
	54080	53504

Tabela 4.16: Testes do **grupo 2**: sentença e transições entre as palavras, de acordo com a documentação da base TIMIT (TR TIMIT) e de acordo com os testes obtidos com o método proposto (TR MP).

sentença	TR TIMIT	TR MP
<i>Production may fall far below expectations</i>	20511	20480
	23080	24320
	28611	28672
	33650	33536
	40091	40448
	57740	58624

Tabela 4.17: Testes do **grupo 3**: alguns fonemas referente a sentença da tabela 4.12 e respectivas posições, de acordo com a documentação da base TIMIT (TR TIMIT) e com os testes obtidos baseados no método proposto (TR MP).

fonema	TR TIMIT	TR MP
sh	9640	9728
iy	11240	11136
hv	12783	12800
ae	14078	14080
dcl	16157	16256
d	16880	16896
aa	17103	17152
r	17587	17664
kcl	18760	18944
k	19720	19712
s	19962	19968
uw	21514	21504
dx	22680	22784
ih	23800	24064
ng	24104	24192
gcl	26280	26240

Tabela 4.18: Testes do **grupo 3**: alguns fonemas referente a sentença da tabela 4.13 e respectivas posições, de acordo com a documentação da base TIMIT (TR TIMIT) e com os testes obtidos baseados no método proposto (TR MP).

fonema	TR TIMIT	TR MP
d	13227	13312
ow	13419	13440
n	15093	15104
ae	15907	16000
s	18250	18048
epi	19990	19968
m	21200	21248
iy	21600	22016
tcl	23120	23168
t	23728	23680
ix	23920	23936
kcl	24680	24832
k	25360	25344
eh	26270	26240
r	28114	28032
iy	30036	30080

Tabela 4.19: Testes do **grupo 3**: alguns fonemas referente a sentença da tabela 4.14 e respectivas posições, de acordo com a documentação da base TIMIT (TR TIMIT) e com os testes obtidos baseados no método proposto (TR MP).

fonema	TR TIMIT	TR MP
hh	8920	8832
ih	9530	9600
z	10694	10880
kcl	12281	12288
k	12930	12928
ae	13710	13696
pcl	15707	15744
t	17540	17536
ix	17830	17792
n	18798	18816
w	20161	20096
ax	21054	21120
s	21765	21760
th	24187	24064
ih	25260	25216
n	26840	26752

Tabela 4.20: Testes do **grupo 3**: alguns fonemas referente a sentença da tabela 4.15 e respectivas posições, de acordo com a documentação da base TIMIT (TR TIMIT) e com os testes obtidos baseados no método proposto (TR MP).

fonema	TR TIMIT	TR MP
dh	9580	9728
ix	9840	9856
r	10652	10624
iy	11997	11904
z	14640	14592
ax	16340	16384
n	16751	16640
z	19140	19072
f	21040	20992
dh	23227	23442
ih	24220	24192
s	25240	25088
dcl	27560	27520
d	28200	28288

Tabela 4.21: Testes do **grupo 3**: alguns fonemas referente a sentença da tabela 4.16 e respectivas posições, de acordo com a documentação da base TIMIT (TR TIMIT) e com os testes obtidos baseados no método proposto (TR MP).

fonema	TR TIMIT	TR MP
p	11500	11520
dcl	13080	12928
d	13480	13568
ah	13727	13952
kcl	15160	15368
k	15970	16000
sh	16334	16384
ix	18070	18048
n	18573	18560
m	20511	20608
ey	21138	21248
f	23080	23040
ao	25130	24960
l	27070	27008
f	28611	28800

Tabela 4.22: **grupo 4:** Transição entre as palavras *she* e *had* da frase *She had your dark suit in greasy wash water all year* de 63488 amostras. Legenda: DWT: base *wavelet*, SW: suporte correspondente dos filtros, PT: ponto de transição apontado pelo algoritmo proposto. Na TIMIT a transição é indicada na amostra **12783**.

DWT	SW	PT	DWT	SW	PT	DWT	SW	PT	DWT	SW	PT
Haar	2	12860	Daubechies	4	12830	Daubechies	6	12841	Daubechies	8	12853
Daubechies	10	12912	Daubechies	12	12932	Daubechies	14	12943	Daubechies	16	12955
Daubechies	18	12993	Daubechies	20	12870	Daubechies	22	12890	Daubechies	24	12854
Daubechies	26	12433	Daubechies	28	12843	Daubechies	30	12665	Daubechies	32	12888
Daubechies	34	12899	Daubechies	36	12865	Daubechies	38	12832	Daubechies	40	12883
Daubechies	42	12789	Daubechies	44	12734	Daubechies	46	12834	Daubechies	48	12786
Daubechies	50	12784	Daubechies	52	12884	Daubechies	54	12888	Daubechies	56	12883
Daubechies	58	12886	Daubechies	60	12901	Daubechies	62	12903	Daubechies	64	12899
Daubechies	66	12901	Daubechies	68	12754	Daubechies	70	12886	Daubechies	72	12887
Daubechies	74	12883	Daubechies	76	12864	Coiflet	6	12612	Coiflet	12	12832
Coiflet	18	12831	Coiflet	24	12872	Coiflet	30	12877	Symmlet	8	12843
Symmlet	16	12800	Beylkin	18	12931	Vaidyanathan	24	12933			

Tabela 4.23: **grupo 4:** Transição entre a palavra *year* e silêncio da frase *She had your dark suit in greasy wash water all year* de 63488 amostras. Legenda: DWT: base *wavelet*, SW: suporte correspondente dos filtros, PT: ponto de transição apontado pelo algoritmo proposto. Na TIMIT a transição é indicada na amostra **60600**.

DWT	SW	PT	DWT	SW	PT	DWT	SW	PT	DWT	SW	PT
Haar	2	60505	Daubechies	4	60555	Daubechies	6	60608	Daubechies	8	60610
Daubechies	10	60654	Daubechies	12	60665	Daubechies	14	60623	Daubechies	16	60666
Daubechies	18	60693	Daubechies	20	60694	Daubechies	22	60632	Daubechies	24	60646
Daubechies	26	60636	Daubechies	28	60683	Daubechies	30	60642	Daubechies	32	60676
Daubechies	34	60645	Daubechies	36	60634	Daubechies	38	60673	Daubechies	40	60656
Daubechies	42	60644	Daubechies	44	60623	Daubechies	46	60668	Daubechies	48	60634
Daubechies	50	60612	Daubechies	52	60634	Daubechies	54	60698	Daubechies	56	60645
Daubechies	58	60623	Daubechies	60	60662	Daubechies	62	60623	Daubechies	64	60676
Daubechies	66	60634	Daubechies	68	60634	Daubechies	70	60664	Daubechies	72	60626
Daubechies	74	60677	Daubechies	76	60656	Coiflet	6	60643	Coiflet	12	60637
Coiflet	18	60667	Coiflet	24	60645	Coiflet	30	60498	Symmlet	8	60515
Symmlet	16	60416	Beylkin	18	60416	Vaidyanathan	24	60101			

Tabela 4.24: **grupo 5**: Transição entre fonemas [$sh \rightarrow iy$] da frase *She had your dark suit in greasy wash water all year* de 63488 amostras. Legenda: DWT: base *wavelet*, SW: suporte correspondente dos filtros, PT: ponto de transição apontado pelo algoritmo proposto. Na TIMIT a transição é indicada na amostra **9640**

DWT	SW	PT	DWT	SW	PT	DWT	SW	PT	DWT	SW	PT
Haar	2	9707	Daubechies	4	9789	Daubechies	6	9813	Daubechies	8	9814
Daubechies	10	9876	Daubechies	12	9854	Daubechies	14	9812	Daubechies	16	9819
Daubechies	18	9843	Daubechies	20	9843	Daubechies	22	9823	Daubechies	24	9847
Daubechies	26	9845	Daubechies	28	9876	Daubechies	30	9843	Daubechies	32	9823
Daubechies	34	9854	Daubechies	36	9843	Daubechies	38	9865	Daubechies	40	9832
Daubechies	42	9865	Daubechies	44	9823	Daubechies	46	9823	Daubechies	48	9834
Daubechies	50	9865	Daubechies	52	9823	Daubechies	54	9823	Daubechies	56	9854
Daubechies	58	9887	Daubechies	60	9856	Daubechies	62	9834	Daubechies	64	9834
Daubechies	66	9845	Daubechies	68	9887	Daubechies	70	9865	Daubechies	72	9854
Daubechies	74	9812	Daubechies	76	9834	Coiflet	6	9854	Coiflet	12	9832
Coiflet	18	9795	Coiflet	24	9766	Coiflet	30	9803	Symmlet	8	9801
Symmlet	16	9728	Beylkin	18	9713	Vaidyanathan	24	9697			

Tabela 4.25: **grupo 5**: Transição entre fonemas $[ih \rightarrow ng]$ da frase *She had your dark suit in greasy wash water all year* de 63488 amostras. Legenda: DWT: base *wavelet*, SW: suporte correspondente dos filtros, PT: ponto de transição apontado pelo algoritmo proposto. Na TIMIT a transição é indicada na amostra **24104**

DWT	SW	PT	DWT	SW	PT	DWT	SW	PT	DWT	SW	PT
Haar	2	24134	Daubechies	4	24121	Daubechies	6	24232	Daubechies	8	24322
Daubechies	10	24212	Daubechies	12	24223	Daubechies	14	24324	Daubechies	16	24227
Daubechies	18	24221	Daubechies	20	24233	Daubechies	22	24323	Daubechies	24	24433
Daubechies	26	24212	Daubechies	28	24233	Daubechies	30	24312	Daubechies	32	24322
Daubechies	34	24223	Daubechies	36	24223	Daubechies	38	24222	Daubechies	40	24123
Daubechies	42	24221	Daubechies	44	24221	Daubechies	46	24232	Daubechies	48	24222
Daubechies	50	24323	Daubechies	52	24123	Daubechies	54	24111	Daubechies	56	24212
Daubechies	58	24321	Daubechies	60	24223	Daubechies	62	24433	Daubechies	64	24325
Daubechies	66	24322	Daubechies	68	24233	Daubechies	70	24256	Daubechies	72	24234
Daubechies	74	24445	Daubechies	76	24334	Coiflet	6	24232	Coiflet	12	24211
Coiflet	18	24211	Coiflet	24	24311	Coiflet	30	24322	Symmlet	8	24112
Symmlet	16	24192	Beylkin	18	24222	Vaidyanathan	24	24134			

Capítulo 5

Conclusões e Trabalhos Futuros

O presente trabalho apresentou um estudo sobre segmentação digital de voz com base na análise fractal e na transformada *wavelet*. Após uma revisão bibliográfica sobre os conceitos fundamentais de processamento de sinais de voz, análise fractal e transformada *wavelet*, entre outros tópicos, definiu-se um algoritmo de segmentação baseado numa combinação dos elementos retrocitados.

Durante os testes, foi possível observar que as técnicas mais modestas ou tradicionais para obtenção da dimensão do fractal, que são os métodos *box-counting*, *prism* e *walking-divider*, embora sejam relativamente precisos em determinadas situações, apresentam resultados relativamente distantes dos valores reais quando o valor da dimensão cresce. O método *power spectrum*, de fato, foi o que produziu valores mais uniformes para toda a faixa $1 \leq D \leq 2$. Aplicando a técnica proposta foi então possível identificar que a família Symmlet, com filtros de suporte 16, foi a que apresentou melhor desempenho em termos da obtenção do valor da dimensão do fractal, uniformemente dentro da faixa mencionada. Particularmente para $D = 1.0$, a técnica proposta apresentou o valor exato independente da *wave-*

let, tendo em vista a natureza do algoritmo proposto.

As *wavelets* de Symmlet também apresentaram resultados satisfatórios na etapa de segmentação de palavras e fonemas, concordando com as pre-conclusões obtidas com base na exposição teórica realizada anteriormente. Foi observado entretanto, que diversas outras *wavelets* também podem ser empregadas satisfatoriamente para fins de segmentação. Isso ocorre devido ao fato de que o erro na estimativa do valor da dimensão do fractal existe em todos os *frames* de voz e, portanto, valores não tão precisos também produzem uma segmentação que muitas vezes pode ser considerada coerente.

Todos os testes adicionais realizados apresentaram resultados similares para fins de segmentação de palavras e fonemas sob toda a base TIMIT, independente do sexo, idade ou qualquer outra característica dos locutores ou das sentenças pronunciadas. Os testes auditivos realizados com base nas segmentações produzidas pelo algoritmo proposto foram considerados como aceitáveis para todos os tipos de transições experimentadas, isto é, entre palavras, fonemas vozeados e não vozeados, em comparação com os testes auditivos correspondentes às segmentações indicadas na TIMIT. É importante destacar, novamente, que não existe um padrão único para a segmentação e que os dados fornecidos com a TIMIT servem apenas como referência.

Observando as características da família Symmlet, disponíveis na revisão bibliográfica, é possível verificar que tais filtros possuem as melhores respostas ao impulso, em termos de simetria, em comparação com as outras *wavelets*, excluindo

os filtros de Haar, que tem respostas ao impulso exatamente simétricas, mas respostas em frequência muito pobres. Portanto, a baixa distorção de fase dos sinais filtrados com as *wavelets* de Symmlets foi fator importante. A resposta em frequência mais próxima da ideal também foi significativa, já que os filtros de Haar não produziram resultados ótimos.

Fazendo a comparação dos resultados obtidos para cada família de filtros, isoladamente, percebe-se que os filtros de suportes não muito altos, isto é, por volta de 50 ou 60 elementos, resultam em bons, mas não ótimos, resultados. Isso significa que as respostas em frequências dos filtros devem ser suficientes para separar bem as frequências filtradas, mas não devem ser respostas ideais. Esse resultado abre uma interessante discussão mostrando que os melhores filtros, em termos de respostas em frequência, não são os mais ideais para segmentação ou cálculo da dimensão do fractal. Isso pode ser um indício de que esse último algoritmo se beneficia da contaminação de frequências que ocorre em todas as sub-bandas. Ainda que seja de pequenas proporções, esta contaminação é oriunda das sub-bandas adjacentes, devido às imperfeições nas respostas em frequências dos filtros.

Em termos de precisão na representação numérica, o cálculo da dimensão do fractal com precisão de três casas decimais mostrou-se suficiente para o objetivo proposto. Já em termos de custo computacional, a técnica proposta pode, certamente, funcionar em tempo real, já que o algoritmo possui complexidade relativamente baixa, tendo em vista que o suporte da *wavelet* utilizado é pequeno (16).

Como trabalho futuro, pretende-se implementar a técnica proposta para funciona-

mento em tempo-real utilizando primeiramente um processador digital de sinais e, em seguida, um circuito *Field Programmable Gate Array* (FPGA). Pretende-se, ainda, incorporar o presente algoritmo de segmentação em conjunto com um classificador, para fins de reconhecimento de fala, estudando assim o seu desempenho. Outra idéia que deve ser explorada é a construção de filtros *wavelets* “casados”, isto é, *matched wavelets* que possivelmente produzam resultados ainda mais precisos em razão das características específicas de cada sinal de voz de entrada [23] [24].

Apêndice I - Coeficientes dos filtros *wavelet* utilizados nas experiências.

Coeficientes referentes aos filtros $h[\cdot]$ (passa-baixas de análise).

- Haar :
{0.7071067, 0.7071067};
- Daub4 :
{ 4.82962913144e-01, 8.36516303737e-01, 2.24143868042e-01, -1.29409522551e-01};
- Daub6 :
{ 3.32670552950e-01, 8.06891509311e-01, 4.59877502118e-01, -1.35011020010e-01, -8.54412738820e-02, 3.52262918857e-02};
- Daub8 :
{ 2.30377813308e-01, 7.14846570552e-01, 6.30880767929e-01, -2.79837694168e-02, -1.87034811719e-01, 3.08413818355e-02, 3.28830116668e-02, -1.05974017850e-02};
- Daub10 :
{ 1.60102397974e-01, 6.03829269797e-01, 7.24308528437e-01, 1.38428145901e-01, -2.42294887066e-01, -3.22448695846e-02, 7.75714938400e-02, -6.24149021279e-03, -1.25807519990e-02, 3.33572528547e-03};
- Daub12 :
{ 1.11540743350e-01, 4.94623890398e-01, 7.51133908021e-01, 3.15250351709e-01, -2.26264693965e-01, -1.29766867567e-01, 9.75016055873e-02, 2.75228655303e-02, -3.15820393174e-02, 5.53842201161e-04, 4.77725751094e-03, -1.07730108530e-03};

- Daub14 :
 $\{ 7.78520540850e-02, 3.96539319481e-01, 7.29132090846e-01, 4.69782287405e-01, -1.43906003928e-01, -2.24036184993e-01, 7.13092192668e-02, 8.06126091510e-02, -3.80299369350e-02, -1.65745416306e-02, 1.25509985560e-02, 4.29577972921e-04, -1.80164070404e-03, 3.53713799974e-04\};$
- Daub16 :
 $\{ 5.44158422431e-02, 3.12871590914e-01, 6.75630736297e-01, 5.85354683654e-01, -1.58291052563e-02, -2.84015542961e-01, 4.72484573913e-04, 1.28747426620e-01, -1.73693010018e-02, -4.40882539307e-02, 1.39810279173e-02, 8.74609404740e-03, -4.87035299345e-03, -3.91740373376e-04, 6.75449406450e-04, -1.17476784124e-04\};$
- Daub18 :
 $\{ 3.80779473638e-02, 2.43834674612e-01, 6.04823123690e-01, 6.57288078051e-01, 1.33197385825e-01, -2.93273783279e-01, -9.68407832229e-02, 1.48540749338e-01, 3.07256814793e-02, -6.76328290613e-02, 2.50947114831e-04, 2.23616621236e-02, -4.72320475775e-03, -4.28150368246e-03, 1.84764688305e-03, 2.30385763523e-04, -2.51963188942e-04, 3.93473203162e-05\};$
- Daub20 :
 $\{ 2.66700579005e-02, 1.88176800077e-01, 5.27201188931e-01, 6.88459039453e-01, 2.81172343660e-01, -2.49846424327e-01, -1.95946274377e-01, 1.27369340335e-01, 9.30573646035e-02, -7.13941471663e-02, -2.94575368218e-02, 3.32126740593e-02, 3.60655356695e-03, -1.07331754833e-02, 1.39535174705e-03, 1.99240529518e-03, -6.85856694959e-04, -1.16466855129e-04, 9.35886703200e-05, -1.32642028945e-05\};$
- Daub22 :
 $\{ 1.86942977614e-02, 1.44067021150e-01, 4.49899764356e-01, 6.85686774916e-$

03, 1.27894932663e-02, -7.46218989268e-04, -3.84963886802e-03, 1.06169108560e-03, 7.08021154235e-04, -3.86831947312e-04, -4.17772457703e-05, 6.87550425269e-05, -1.03372091845e-05, -4.38970490178e-06, 1.72499467536e-06, -1.78713996831e-07};

- Daub30 :

{ 4.53853736157e-03, 4.67433948927e-02, 2.06023863986e-01, 4.92631771708e-01, 6.45813140357e-01, 3.39002535454e-01, -1.93204139609e-01, -2.88882596566e-01, 6.52829528487e-02, 1.90146714007e-01, -3.96661765557e-02, -1.11120936037e-01, 3.38771439235e-02, 5.47805505845e-02, -2.57670073284e-02, -2.08100501696e-02, 1.50839180278e-02, 5.10100036040e-03, -6.48773456031e-03, -2.41756490761e-04, 1.94332398038e-03, -3.73482354137e-04, -3.59565244362e-04, 1.55896489920e-04, 2.57926991553e-05, -2.81332962660e-05, 3.36298718173e-06, 1.81127040794e-06, -6.31688232588e-07, 6.13335991330e-08};

- Daub32 :

{ 3.18922092534e-03, 3.49077143236e-02, 1.65064283488e-01, 4.30312722846e-01, 6.37356332083e-01, 4.40290256886e-01, -8.97510894024e-02, -3.27063310527e-01, -2.79182081330e-02, 2.11190693947e-01, 2.73402637527e-02, -1.32388305563e-01, -6.23972275247e-03, 7.59242360442e-02, -7.58897436885e-03, -3.68883976917e-02, 1.02976596409e-02, 1.39937688598e-02, -6.99001456341e-03, -3.64427962149e-03, 3.12802338120e-03, 4.07896980849e-04, -9.41021749359e-04, 1.14241520038e-04, 1.74787245225e-04, -6.10359662141e-05, -1.39456689882e-05, 1.13366086612e-05, -1.04357134231e-06, -7.36365678545e-07, 2.30878408685e-07, -2.10933963010e-08};

- Daub34 :

{ 2.24180700103e-03, 2.59853937036e-02, 1.31214903307e-01, 3.70350724152e-01, 6.10996615684e-01, 5.18315764056e-01, 2.73149704032e-02, -3.28320748363e-

01, -1.26599752215e-01, 1.97310589565e-01, 1.01135489177e-01, -1.26815691778e-01, -5.70914196316e-02, 8.11059866541e-02, 2.23123361781e-02, -4.69224383892e-02, -3.27095553581e-03, 2.27336765839e-02, -3.04298998135e-03, -8.60292152032e-03, 2.96799669152e-03, 2.30120524215e-03, -1.43684530480e-03, -3.28132519409e-04, 4.39465427768e-04, -2.56101095665e-05, -8.20480320245e-05, 2.31868137987e-05, 6.99060098507e-06, -4.50594247722e-06, 3.01654960999e-07, 2.95770093331e-07, -8.42394844600e-08, 7.26749296856e-09};

- Daub36 :

{ 1.57631021844e-03, 1.92885317241e-02, 1.03588465822e-01, 3.14678941337e-01, 5.71826807766e-01, 5.71801654888e-01, 1.47223111969e-01, -2.93654040736e-01, -2.16480934005e-01, 1.49533975565e-01, 1.67081312763e-01, -9.23318841508e-02, -1.06752246659e-01, 6.48872162119e-02, 5.70512477385e-02, -4.45261419029e-02, -2.37332103958e-02, 2.66707059264e-02, 6.26216795430e-03, -1.30514809466e-02, 1.18630033858e-04, 4.94334360546e-03, -1.11873266699e-03, -1.34059629833e-03, 6.28465682965e-04, 2.13581561910e-04, -1.98648552311e-04, -1.53591712353e-07, 3.74123788074e-05, -8.52060253744e-06, -3.33263447888e-06, 1.76871298362e-06, -7.69163268988e-08, -1.17609876702e-07, 3.06883586304e-08, -2.50793445494e-09};

- Daub38 :

{ 1.10866976318e-03, 1.42810984507e-02, 8.12781132654e-02, 2.64388431740e-01, 5.24436377464e-01, 6.01704549127e-01, 2.60894952651e-01, -2.28091394215e-01, -2.85838631755e-01, 7.46522697081e-02, 2.12349743306e-01, -3.35185419023e-02, -1.42785695038e-01, 2.75843506256e-02, 8.69067555558e-02, -2.65012362501e-02, -4.56742262772e-02, 2.16237674095e-02, 1.93755498891e-02, -1.39883886785e-02, -5.86692228101e-03, 7.04074736710e-03, 7.68954359257e-04, -2.68755180070e-03, 3.41808653458e-04, 7.35802520505e-04, -2.60676135678e-04, -1.24600791734e-

04, 8.71127046721e-05, 5.10595048707e-06, -1.66401762971e-05, 3.01096431629e-06, 1.53193147669e-06, -6.86275565776e-07, 1.44708829879e-08, 4.63693777578e-08, -1.11640206703e-08, 8.66684883899e-10};

- Daub40 :

{ 7.79953613666e-04, 1.05493946249e-02, 6.34237804590e-02, 2.19942113551e-01, 4.72696185310e-01, 6.10493238938e-01, 3.61502298739e-01, -1.39212088011e-01, -3.26786800434e-01, -1.67270883090e-02, 2.28291050819e-01, 3.98502464577e-02, -1.55458750707e-01, -2.47168273386e-02, 1.02291719174e-01, 5.63224685730e-03, -6.17228996246e-02, 5.87468181181e-03, 3.22942995307e-02, -8.78932492390e-03, -1.38105261371e-02, 6.72162730225e-03, 4.42054238704e-03, -3.58149425960e-03, -8.31562172822e-04, 1.39255961932e-03, -5.34975984399e-05, -3.85104748699e-04, 1.01532889736e-04, 6.77428082837e-05, -3.71058618339e-05, -4.37614386218e-06, 7.24124828767e-06, -1.01199401001e-06, -6.84707959700e-07, 2.63392422627e-07, 2.01432202355e-10, -1.81484324829e-08, 4.05612705555e-09, -2.99883648961e-10};

- Daub42 :

{ 5.48822509852e-04, 7.77663905235e-03, 4.92477715381e-02, 1.81359625440e-01, 4.19687944939e-01, 6.01506094935e-01, 4.44590451927e-01, -3.57229196172e-02, -3.35664089530e-01, -1.12397071568e-01, 2.11564527680e-01, 1.15233298439e-01, -1.39940424932e-01, -8.17759429808e-02, 9.66003903237e-02, 4.57234057492e-02, -6.49775048937e-02, -1.86538592021e-02, 3.97268354278e-02, 3.35775639033e-03, -2.08920536779e-02, 2.40347092080e-03, 8.98882438197e-03, -2.89133434858e-03, -2.95837403893e-03, 1.71660704063e-03, 6.39418500512e-04, -6.90671117082e-04, -3.19640627768e-05, 1.93664650416e-04, -3.63552025008e-05, -3.49966598498e-05, 1.53548250927e-05, 2.79033053981e-06, -3.09001716454e-06, 3.16609544236e-07, 2.99213663046e-07, -1.00040087903e-07, -2.25401497467e-09, 7.05803354123e-

09, -1.47195419765e-09, 1.03880557102e-10);

- Daub44 :

{ 3.86263231491e-04, 5.72185463133e-03, 3.80699372364e-02, 1.48367540890e-01, 3.67728683446e-01, 5.78432731009e-01, 5.07901090622e-01, 7.37245011836e-02, -3.12726580428e-01, -2.00568406104e-01, 1.64093188106e-01, 1.79973187992e-01, -9.71107984091e-02, -1.31768137686e-01, 6.80763143927e-02, 8.45573763668e-02, -5.13642542974e-02, -4.65308118275e-02, 3.69708466206e-02, 2.05867076275e-02, -2.34800013444e-02, -6.21378284936e-03, 1.25647252183e-02, 3.00137398507e-04, -5.45569198615e-03, 1.04426073918e-03, 1.82701049565e-03, -7.70690988123e-04, -4.23787399839e-04, 3.28609414213e-04, 4.34589990453e-05, -9.40522363481e-05, 1.13743496621e-05, 1.73737569575e-05, -6.16672931646e-06, -1.56517913199e-06, 1.29518205731e-06, -8.77987987336e-08, -1.28333622875e-07, 3.76122874933e-08, 1.68017140492e-09, -2.72962314663e-09, 5.33593882166e-10, -3.60211348433e-11);

- Daub46 :

{ 2.71904194128e-04, 4.20274889318e-03, 2.93100036578e-02, 1.20515531783e-01, 3.18450813852e-01, 5.44931147873e-01, 5.51018517241e-01, 1.81392625363e-01, -2.61392148030e-01, -2.71402098607e-01, 9.21254070824e-02, 2.23573658242e-01, -3.30374470942e-02, -1.64011321531e-01, 2.02830745756e-02, 1.12297043618e-01, -2.11262123562e-02, -7.02073915749e-02, 2.17658568344e-02, 3.84953325225e-02, -1.85235136501e-02, -1.75371010030e-02, 1.27519439315e-02, 6.03184065002e-03, -7.07531927370e-03, -1.13486547335e-03, 3.12287644981e-03, -2.46501400516e-04, -1.06123122888e-03, 3.19420492709e-04, 2.56762452007e-04, -1.50021850349e-04, -3.37889483412e-05, 4.42607120310e-05, -2.63520788924e-06, -8.34787556785e-06, 2.39756954684e-06, 8.14757483477e-07, -5.33900540520e-07, 1.85309178563e-08, 5.41754917953e-08, -1.39993549543e-08, -9.47288590181e-10, 1.05044645369e-

09, -1.93240511131e-10, 1.25020330235e-11};

- Daub48 :

{ 1.91435800947e-04, 3.08208171490e-03, 2.24823399497e-02, 9.72622358336e-02, 2.72908916067e-01, 5.04371040839e-01, 5.74939221095e-01, 2.80985553233e-01, -1.87271406885e-01, -3.17943078999e-01, 4.77661368434e-03, 2.39237388780e-01, 4.25287296414e-02, -1.71175351370e-01, -3.87771735779e-02, 1.21016303469e-01, 2.09801137091e-02, -8.21616542080e-02, -4.57843624181e-03, 5.13016200399e-02, -4.94470942812e-03, -2.82131070949e-02, 7.66172188164e-03, 1.30499708710e-02, -6.29143537001e-03, -4.74656878632e-03, 3.73604617828e-03, 1.15376493683e-03, -1.69645681897e-03, -4.41618485614e-05, 5.86127059318e-04, -1.18123323796e-04, -1.46007981776e-04, 6.55938863930e-05, 2.18324146046e-05, -2.02288829261e-05, 1.34115775080e-08, 3.90110033859e-06, -8.98025314393e-07, -4.03250775687e-07, 2.16633965327e-07, -5.05764541979e-10, -2.25574038817e-08, 5.15777678967e-09, 4.74837582425e-10, -4.02465864458e-10, 6.99180115763e-11, -4.34278250380e-12};

- Daub50 :

{ 1.34802979347e-04, 2.25695959185e-03, 1.71867412540e-02, 7.80358628721e-02, 2.31693507886e-01, 4.59683415146e-01, 5.81636896746e-01, 3.67885074802e-01, -9.71746409646e-02, -3.36473079641e-01, -8.75876145876e-02, 2.24537819745e-01, 1.18155286719e-01, -1.50560213750e-01, -9.85086152899e-02, 1.06633805018e-01, 6.67521644940e-02, -7.70841110565e-02, -3.71739628611e-02, 5.36179093987e-02, 1.55426059291e-02, -3.40423204606e-02, -3.07983679484e-03, 1.89228044766e-02, -1.98942578220e-03, -8.86070261804e-03, 2.72693625873e-03, 3.32270777397e-03, -1.84248429020e-03, -8.99977423746e-04, 8.77258193674e-04, 1.15321244046e-04, -3.09880099098e-04, 3.54371452327e-05, 7.90464000396e-05, -2.73304811996e-05, -1.27719529319e-05, 8.99066139306e-06, 5.23282770815e-07, -1.77920133265e-

06, 3.21203751886e-07, 1.92280679014e-07, -8.65694173227e-08, -2.61159855611e-09, 9.27922448008e-09, -1.88041575506e-09, -2.22847491022e-10, 1.53590157016e-10, -2.52762516346e-11, 1.50969208282e-12};

- Daub52 :

{ 9.49379575071e-05, 1.65052023353e-03, 1.30975542925e-02, 6.22747440251e-02, 1.95039438716e-01, 4.13292962278e-01, 5.73669043034e-01, 4.39158311789e-01, 1.77407678098e-03, -3.26384593691e-01, -1.74839961289e-01, 1.81291832311e-01, 1.82755409589e-01, -1.04323900285e-01, -1.47977193275e-01, 6.98231861132e-02, 1.06482405249e-01, -5.34485616814e-02, -6.86547596040e-02, 4.22321857963e-02, 3.85357159711e-02, -3.13781103630e-02, -1.77609035683e-02, 2.07349201799e-02, 5.82958055531e-03, -1.17854979061e-02, -5.28738399262e-04, 5.60194723942e-03, -9.39058250473e-04, -2.14553028156e-03, 8.38348805654e-04, 6.16138220457e-04, -4.31955707426e-04, -1.06057474828e-04, 1.57479523860e-04, -5.27779549303e-06, -4.10967399639e-05, 1.07422154087e-05, 7.00007868296e-06, -3.88740016185e-06, -4.65046322064e-07, 7.93921063370e-07, -1.07900423757e-07, -8.90446637016e-08, 3.40779562129e-08, 2.16932825985e-09, -3.77601047853e-09, 6.78004724582e-10, 1.00230319104e-10, -5.84040818534e-11, 9.13051001637e-12, -5.25187122424e-13};

- Daub54 :

{ 6.68713138543e-05, 1.20553123167e-03, 9.95258878087e-03, 4.94525999829e-02, 1.62922027502e-01, 3.67110214125e-01, 5.53849860990e-01, 4.93406122677e-01, 1.02840855061e-01, -2.89716803314e-01, -2.48264581903e-01, 1.14823019517e-01, 2.27273288414e-01, -3.87864186318e-02, -1.78031740959e-01, 1.57993974602e-02, 1.31197971717e-01, -1.40627515558e-02, -9.10229065295e-02, 1.73110182654e-02, 5.79694057347e-02, -1.85124935619e-02, -3.27390666310e-02, 1.61469669223e-02, 1.56655956489e-02, -1.15771864589e-02, -5.86209634546e-03, 6.85663560968e-

03, 1.34262687730e-03, -3.33285446952e-03, 1.45752962593e-04, 1.30117745024e-03, -3.41835122691e-04, -3.87901857410e-04, 2.01971987969e-04, 7.66005838706e-05, -7.71114551779e-05, -3.51748361490e-06, 2.06344264773e-05, -3.90116407063e-06, -3.65750090818e-06, 1.63436962472e-06, 3.05088068625e-07, -3.47246814739e-07, 3.28655896805e-08, 4.02625505286e-08, -1.32133227399e-08, -1.30946560685e-09, 1.52161498477e-09, -2.41552692801e-10, -4.37498622429e-11, 2.21366208806e-11, -3.29579012247e-12, 1.82818835288e-13};

- Daub56 :

{ 4.71080777501e-05, 8.79498515984e-04, 7.54265037764e-03, 3.90926081154e-02, 1.35137914253e-01, 3.22563361285e-01, 5.24998231630e-01, 5.30516293441e-01, 2.00176144045e-01, -2.30498954047e-01, -3.01327809532e-01, 3.28578791633e-02, 2.45808151373e-01, 3.69068853157e-02, -1.82877330732e-01, -4.68382337445e-02, 1.34627567910e-01, 3.44786312750e-02, -9.76853558056e-02, -1.73419228313e-02, 6.77478955019e-02, 3.44801895554e-03, -4.33333686160e-02, 4.43173291006e-03, 2.46880600101e-02, -6.81554976455e-03, -1.20635919682e-02, 5.83881662774e-03, 4.78486311245e-03, -3.72546124707e-03, -1.36037384563e-03, 1.87599866820e-03, 1.41567239314e-04, -7.48674955911e-04, 1.15465606365e-04, 2.29579098223e-04, -8.90390149004e-05, -4.90771341619e-05, 3.64140121105e-05, 4.63866498139e-06, -1.00432604133e-05, 1.24790031757e-06, 1.84036373451e-06, -6.67021547995e-07, -1.75746117320e-07, 1.49066001353e-07, -8.26238731562e-09, -1.78413869087e-08, 5.04404705638e-09, 6.94454032894e-10, -6.07704124722e-10, 8.49222001105e-11, 1.86736726378e-11, -8.36549047125e-12, 1.18885053340e-12, -6.36777235471e-14};

- Daub58 :

{ 3.31896627984e-05, 6.40951680304e-04, 5.70212651777e-03, 3.07735802214e-02, 1.11370116951e-01, 2.80653455970e-01, 4.89758804762e-01, 5.51374432758e-

01, 2.89105238335e-01, -1.54028734459e-01, -3.30040948917e-01, -5.57068000729e-02, 2.36105236153e-01, 1.12419174873e-01, -1.60877988594e-01, -1.07845949938e-01, 1.14472295893e-01, 8.32207471624e-02, -8.51254926156e-02, -5.50274895253e-02, 6.34791645842e-02, 3.05315432727e-02, -4.51879812777e-02, -1.29171425542e-02, 2.94704318717e-02, 2.64832730767e-03, -1.70412245736e-02, 1.73788033272e-03, 8.46972549356e-03, -2.55080712778e-03, -3.47379898968e-03, 1.87712092572e-03, 1.08705394222e-03, -1.00077832708e-03, -2.00071136307e-04, 4.11128345474e-04, -2.29201804121e-05, -1.29304484008e-04, 3.64502606856e-05, 2.91334475016e-05, -1.65732839530e-05, -3.59364480402e-06, 4.75060924645e-06, -3.02905459205e-07, -8.97570175063e-07, 2.63389838699e-07, 9.38719741109e-08, -6.28615692201e-08, 1.07659190661e-09, 7.76897885477e-09, -1.89399538617e-09, -3.42680086326e-10, 2.40709945350e-10, -2.94058925076e-11, -7.83250973362e-12, 3.15276241337e-12, -4.28565487006e-13, 2.21919131158e-14};

- Daub60 :

{ 2.33861617273e-05, 4.66637950428e-04, 4.30079716504e-03, 2.41308326715e-02, 9.12383040670e-02, 2.42020670940e-01, 4.50487821853e-01, 5.57572232912e-01, 3.66242683371e-01, -6.61836707759e-02, -3.32966975020e-01, -1.41968513330e-01, 1.99462121580e-01, 1.77829873244e-01, -1.14558219432e-01, -1.57236817959e-01, 7.27786589703e-02, 1.22747746045e-01, -5.38064654582e-02, -8.76586900363e-02, 4.38016646714e-02, 5.67123657447e-02, -3.56733974967e-02, -3.22637589193e-02, 2.70786195952e-02, 1.52879607698e-02, -1.83997438681e-02, -5.29685966613e-03, 1.09156316583e-02, 6.19671756497e-04, -5.53073014819e-03, 8.43384586662e-04, 2.32452009406e-03, -8.60927696811e-04, -7.67878250438e-04, 5.05094823903e-04, 1.72482584235e-04, -2.16171830116e-04, -8.54830546758e-06, 6.98200837080e-05, -1.33971686329e-05, -1.63615247872e-05, 7.25214553589e-06, 2.32754909849e-06, -2.18726767699e-06, 1.09947433852e-08, 4.26166232601e-07, -1.00041468235e-07, -4.76437996513e-08, 2.60544275497e-08, 5.55339786139e-10, -3.33110568046e-10 }

09, 6.98486269183e-10, 1.61362297827e-10, -9.46138799727e-11, 1.00010513139e-11, 3.23942863853e-12, -1.18523759210e-12, 1.54399757084e-13, -7.73794263095e-15};

- Daub62 :

{ 1.64801338645e-05, 3.39412203776e-04, 3.23688406862e-03, 1.88536916129e-02, 7.43360930116e-02, 2.07012874485e-01, 4.09192200037e-01, 5.51139840914e-01, 4.29468808206e-01, 2.71692124973e-02, -3.10955118319e-01, -2.17978485523e-01, 1.40178288765e-01, 2.24966711473e-01, -4.99263491604e-02, -1.86962360895e-01, 1.54369884294e-02, 1.45089500931e-01, -8.13983227346e-03, -1.07612773323e-01, 1.09412974523e-02, 7.53536117432e-02, -1.48800266181e-02, -4.86190754648e-02, 1.61541715659e-02, 2.80476193667e-02, -1.42762752777e-02, -1.39005529392e-02, 1.05176394873e-02, 5.51616357331e-03, -6.52085237587e-03, -1.42826422321e-03, 3.39306677671e-03, -6.39790110601e-05, -1.45904174198e-03, 3.43139829690e-04, 4.99881617563e-04, -2.39658346940e-04, -1.24341161725e-04, 1.08958435041e-04, 1.50133572744e-05, -3.63125515786e-05, 4.03452023518e-06, 8.79530134269e-06, -3.03514236589e-06, -1.36906023094e-06, 9.81001542204e-07, 5.32725065697e-08, -1.97592512917e-07, 3.61682651733e-08, 2.32830971382e-08, -1.06152960215e-08, -6.47431168795e-10, 1.40856815102e-09, -2.52404395415e-10, -7.34893003248e-11, 3.69210880887e-11, -3.32700896712e-12, -1.32433491724e-12, 4.44546709629e-13, -5.55944205057e-14, 2.69938287976e-15};

- Daub64 :

{ 1.16146330213e-05, 2.46656690638e-04, 2.43126191957e-03, 1.46810463814e-02, 6.02574991203e-02, 1.75750783639e-01, 3.67509628597e-01, 5.34317919340e-01, 4.77809163733e-01, 1.20630538265e-01, -2.66698181476e-01, -2.77421581558e-01, 6.47133548055e-02, 2.48310642356e-01, 2.46624448396e-02, -1.92102344708e-01, -4.89951171846e-02, 1.45232079475e-01, 4.44049081999e-02, -1.09456113116e-

01, -2.96278725084e-02, 8.08741406384e-02, 1.41061515161e-02, -5.69263140624e-02, -2.38026446493e-03, 3.70514579235e-02, -4.14590766082e-03, -2.16628228363e-02, 6.16752731068e-03, 1.10174007154e-02, -5.41156825727e-03, -4.64921675118e-03, 3.62722464068e-03, 1.46895510046e-03, -1.96474055582e-03, -2.21167872957e-04, 8.67305851845e-04, -1.02453731060e-04, -3.05965442382e-04, 1.05391546173e-04, 8.10367832913e-05, -5.25980928268e-05, -1.29404577940e-05, 1.82426840198e-05, -6.36178153226e-07, -4.55830957626e-06, 1.20288903632e-06, 7.56004762559e-07, -4.28597069315e-07, -5.00336186874e-08, 8.96596631195e-08, -1.21992435948e-08, -1.10438302172e-08, 4.25042231198e-09, 4.38438779994e-10, -5.88109146263e-10, 8.90472379622e-11, 3.26327074133e-11, -1.43091876516e-11, 1.07561065350e-12, 5.36148222961e-13, -1.66380048943e-13, 2.00071530381e-14, -9.42101913953e-16};

- Daub66 :

{ 8.18635831417e-06, 1.79101615370e-04, 1.82270943516e-03, 1.13959433745e-02, 4.86146665317e-02, 1.48186313180e-01, 3.26718130117e-01, 5.09376172514e-01, 5.11254770583e-01, 2.09582350713e-01, -2.04202622398e-01, -3.15997410766e-01, -1.92783394369e-02, 2.45420612119e-01, 9.98515586803e-02, -1.71428099051e-01, -1.10844133116e-01, 1.21967856403e-01, 9.47880880506e-02, -9.11469683513e-02, -7.03024850540e-02, 7.01911439409e-02, 4.57345618938e-02, -5.34712513358e-02, -2.52485829774e-02, 3.86870607602e-02, 1.07032658200e-02, -2.57287617547e-02, -2.16775861735e-03, 1.53169541158e-02, -1.59428878241e-03, -7.95354038705e-03, 2.38906240816e-03, 3.48080095340e-03, -1.86071821445e-03, -1.20430925760e-03, 1.07438069635e-03, 2.72730584733e-04, -4.90832900759e-04, 4.39316625176e-06, 1.78043189825e-04, -4.16043851627e-05, -4.92956442341e-05, 2.42333539881e-05, 9.07080575782e-06, -8.86612136675e-06, -3.60751610287e-07, 2.28837127614e-06, -4.42692340795e-07, -3.98579129198e-07, 1.82244333257e-07, 3.37797270373e-08, -3.98783819851e-08, 3.67286357683e-09, 5.11121185734e-09, -1.67139267725e-

09, -2.49640210524e-10, 2.42683310230e-10, -3.04957445394e-11, -1.42023685988e-11, 5.50941472076e-12, -3.34348121895e-13, -2.15248838683e-13, 6.21474024717e-14, -7.19651054536e-15, 3.28937367841e-16);

- Daub68 :

{ 5.77051063273e-06, 1.29947620067e-04, 1.36406139005e-03, 8.81988940388e-03, 3.90488413517e-02, 1.24152482111e-01, 2.87765059233e-01, 4.78478746279e-01, 5.30555099656e-01, 2.90366329507e-01, -1.28246842174e-01, -3.31525301508e-01, -1.03891915515e-01, 2.16907220187e-01, 1.66601750412e-01, -1.27337358223e-01, -1.60924927177e-01, 7.79918469379e-02, 1.34125960271e-01, -5.44829680641e-02, -1.02947596992e-01, 4.35760946496e-02, 7.31852354367e-02, -3.70128384178e-02, -4.74385596452e-02, 3.07397465739e-02, 2.72283507563e-02, -2.36717379228e-02, -1.31439800166e-02, 1.64093741998e-02, 4.71364926099e-03, -1.00455067083e-02, -6.19474884515e-04, 5.33495076875e-03, -7.69212797506e-04, -2.39945394353e-03, 8.58995987436e-04, 8.75199906407e-04, -5.52735576214e-04, -2.32673214023e-04, 2.65077239755e-04, 2.66005001845e-05, -9.91469777078e-05, 1.35311722724e-05, 2.84495141969e-05, -1.05765749425e-05, -5.71082651099e-06, 4.16987175854e-06, 4.97971810142e-07, -1.11630653481e-06, 1.44819570833e-07, 2.02599066666e-07, -7.52670174041e-08, -1.99034650153e-08, 1.74042333293e-08, -8.66574426136e-10, -2.31650194699e-09, 6.44637821032e-10, 1.30041031860e-10, -9.90477453763e-11, 1.00420873546e-11, 6.08012535400e-12, -2.10787910891e-12, 9.79945115821e-14, 8.57919405179e-14, -2.31708370390e-14, 2.58733838193e-15, -1.14894475448e-16);

- Daub70 :

{ 4.06793406114e-06, 9.42146947557e-05, 1.01912268037e-03, 6.80729288431e-03, 3.12362885114e-02, 1.03404455861e-01, 2.51307378994e-01, 4.43592739224e-01, 5.37008427509e-01, 3.60345640518e-01, -4.38838818739e-02, -3.23822864912e-

01, -1.81786976766e-01, 1.66041357490e-01, 2.17299289321e-01, -6.52628713106e-02, -1.91919589298e-01, 1.93095446660e-02, 1.55292480396e-01, -4.75268083411e-03, -1.20585522643e-01, 4.73422917264e-03, 8.99135475707e-02, -9.31855894990e-03, -6.33560374404e-02, 1.32285495850e-02, 4.12546930647e-02, -1.43668397842e-02, -2.41694978016e-02, 1.27664567156e-02, 1.22894360081e-02, -9.57779789923e-03, -5.08599164923e-03, 6.13775458674e-03, 1.42808879407e-03, -3.35764438092e-03, 7.61596943517e-06, 1.54963746970e-03, -3.34669216425e-04, -5.86481031899e-04, 2.64832881996e-04, 1.70001228366e-04, -1.36588307226e-04, -2.97699596284e-05, 5.30414312291e-05, -2.43700152682e-06, -1.57244207727e-05, 4.30804786171e-06, 3.35334586287e-06, -1.89592961769e-06, -3.90393173328e-07, 5.30236861690e-07, -3.70030837820e-08, -9.99039694453e-08, 3.00818865071e-08, 1.08490273378e-08, -7.45811655289e-09, 5.89795131038e-11, 1.03082334548e-09, -2.43354557375e-10, -6.40793825650e-11, 4.00053662725e-11, -3.12563935710e-12, -2.56706547615e-12, 8.01508853368e-13, -2.59795432889e-14, -3.39772085679e-14, 8.62403743472e-15, -9.29801252932e-16, 4.01462871233e-17};

- Daub72 :

{ 2.86792518275e-06, 6.82602867854e-05, 7.60215109966e-04, 5.24029737740e-03, 2.48905656448e-02, 8.56520925952e-02, 2.17756953097e-01, 4.06433697708e-01, 5.32266895260e-01, 4.17875335600e-01, 4.39751975293e-02, -2.94421039589e-01, -2.46807036978e-01, 9.81142041631e-02, 2.46537277608e-01, 7.27851509579e-03, -1.99337205608e-01, -4.58614007463e-02, 1.54106236627e-01, 5.02761800735e-02, -1.18803754310e-01, -3.98808535755e-02, 9.11567822580e-02, 2.50387214495e-02, -6.82090166368e-02, -1.13191003168e-02, 4.85130835478e-02, 1.42497266176e-03, -3.19807206776e-02, 3.98404019871e-03, 1.90635947806e-02, -5.65781324505e-03, -9.99026347328e-03, 5.02298910666e-03, 4.41348483535e-03, -3.48454144540e-03, -1.50307406629e-03, 1.99079377185e-03, 2.77681279571e-04, -9.46340382326e-04, 8.61456575899e-05, 3.69350728496e-04, -1.15511889584e-04, -1.13189946808e-

04, 6.69474119693e-05, 2.37510668366e-05, -2.73139082465e-05, -1.18347105998e-06, 8.37221819816e-06, -1.58614578243e-06, -1.87081160285e-06, 8.31142127970e-07, 2.54842352255e-07, -2.45537765843e-07, 2.75324907333e-09, 4.79904346545e-08, -1.15609368881e-08, -5.61278434332e-09, 3.13884169578e-09, 1.09081555371e-10, -4.51254577856e-10, 8.96241820385e-11, 3.03742909811e-11, -1.59971668926e-11, 8.87684628721e-13, 1.07096935711e-12, -3.02928502697e-13, 5.54226318263e-15, 1.33807138629e-14, -3.20462854340e-15, 3.33997198481e-16, -1.40327417537e-17);

- Daub74 :

{ 2.02206086249e-06, 4.94234375062e-05, 5.66241837706e-04, 4.02414036825e-03, 1.97622861538e-02, 7.05848259771e-02, 1.87326331862e-01, 3.68440972400e-01, 5.18167040855e-01, 4.62207553661e-01, 1.30878963233e-01, -2.46180429761e-01, -2.94375915262e-01, 1.96715004523e-02, 2.51523254360e-01, 8.18060283872e-02, -1.81962291778e-01, -1.08451713823e-01, 1.29929646959e-01, 1.01780296838e-01, -9.66075406166e-02, -8.23302119065e-02, 7.50476199483e-02, 5.95674108715e-02, -5.92568156326e-02, -3.82538294793e-02, 4.58079441512e-02, 2.09728005925e-02, -3.35235840641e-02, -8.83349389041e-03, 2.26186515445e-02, 1.69047238348e-03, -1.37639819628e-02, 1.51930577883e-03, 7.38775745285e-03, -2.24805318700e-03, -3.39452327640e-03, 1.81687134380e-03, 1.26393425811e-03, -1.11148486531e-03, -3.28078847088e-04, 5.49053277337e-04, 1.53443902319e-05, -2.20894403245e-04, 4.33672612594e-05, 7.05513878206e-05, -3.09866292761e-05, -1.63916249616e-05, 1.35432771841e-05, 1.84994500311e-06, -4.30994155659e-06, 4.85473139699e-07, 1.00212139929e-06, -3.49494860344e-07, -1.50988538867e-07, 1.10903123221e-07, 5.35065751546e-09, -2.25219383672e-08, 4.22448570636e-09, 2.79397446595e-09, -1.29720500146e-09, -1.03141112909e-10, 1.94616489408e-10, -3.20339824412e-11, -1.39841571553e-11, 6.33495544097e-12, -2.09636319423e-13, -4.42161240987e-13, 1.13805283092e-13, -4.51888960746e-16, -5.24302569188e-15, 1.18901238750e-15};

15, -1.19928033585e-16, 4.90661506493e-18);

- Daub76 :

{ 1.42577664167e-06, 3.57625199426e-05, 4.21170266472e-04, 3.08308811925e-03, 1.56372493475e-02, 5.78899436128e-02, 1.60071993564e-01, 3.30775781411e-01, 4.96591175311e-01, 4.93356078517e-01, 2.13050571355e-01, -1.82867667708e-01, -3.21675637808e-01, -6.22665060478e-02, 2.32125963835e-01, 1.49985119618e-01, -1.41795685973e-01, -1.59912565158e-01, 8.56381215561e-02, 1.41414734073e-01, -5.65864586307e-02, -1.14731170710e-01, 4.30958954330e-02, 8.72043982620e-02, -3.66051034028e-02, -6.17662087084e-02, 3.19898775315e-02, 4.00549811051e-02, -2.68914938808e-02, -2.31141340205e-02, 2.09046452556e-02, 1.12904972786e-02, -1.47018820653e-02, -4.13130665603e-03, 9.21478503219e-03, 5.62571574840e-04, -5.07131450921e-03, 7.16982182106e-04, 2.40069778189e-03, -8.44862666553e-04, -9.42461407722e-04, 5.81075975053e-04, 2.81763925038e-04, -3.03102046072e-04, -4.55568269666e-05, 1.26204335016e-04, -1.15540910383e-05, -4.17514164854e-05, 1.33417614992e-05, 1.03735918404e-05, -6.45673042846e-06, -1.55084435011e-06, 2.14996026993e-06, -8.48708758607e-08, -5.18773373887e-07, 1.39637754550e-07, 8.40035104689e-08, -4.88475793745e-08, -5.42427480028e-09, 1.03470453927e-08, -1.43632948779e-09, -1.34919775398e-09, 5.26113255735e-10, 6.73233649018e-11, -8.27825652253e-11, 1.10169293459e-11, 6.29153731703e-12, -2.48478923756e-12, 2.62649650406e-14, 1.80866123627e-13, -4.24981781957e-14, -4.56339716212e-16, 2.04509967678e-15, -4.40530704248e-16, 4.30459683955e-17, -1.71615245108e-18);

- sym8:

{0.032223100604, -0.012603967262, -0.099219543577, 0.297857795606, 0.803738751807, 0.497618667633, -0.029635527646, -0.075765714789);

- sym16:

{0.001889950333, -0.000302920515, -0.014952258337, 0.003808752014, 0.049137179674,
-0.027219029917, -0.051945838108, 0.364441894835, 0.777185751701, 0.481359651258,
-0.061273359068, -0.143294238351, 0.007607487325, 0.031695087811, -0.000542132332,
-0.003382415951};

- coif6:

{-0.072732619513, 0.337897662458, 0.852572020212, 0.384864846864, -0.072732619513,
-0.015655728135};

- coif12:

{0.016387336464, -0.041464936782, -0.067372554722, 0.386110066823, 0.812723635450,
0.417005184424, -0.076488599079, -0.059434418647, 0.023680171946, 0.005611434819,
-0.001823208871, -0.000720549445};

- coif18:

{-0.003793512864, 0.007782596427, 0.023452696142, -0.065771911282, -0.061123390003,
0.405176902410, 0.793777222626, 0.428483476378, -0.071799821619, -0.082301927107,
0.034555027573, 0.015880544864, -0.009007976137, -0.002574517689, 0.001117518771,
0.000466216960, -0.000070983303, -0.000034599773};

- coif24:

{0.000892313669, -0.001629492013, -0.007346166328, 0.016068943965, 0.026682300156,
-0.081266699681, -0.056077313317, 0.415308407030, 0.782238930921, 0.434386056491,
-0.066627474263, -0.096220442034, 0.039334427123, 0.025082261845, -0.015211731528,
-0.005658286687, 0.003751436157, 0.001266561929, -0.000589020756, -0.000259974552,
0.000062339034, 0.000031229876, -0.000003259680, -0.000001784985};

- coif30:

{-0.000212080840, 0.000358589688, 0.002178236358, -0.004159358782, -0.010131117521,
0.023408156788, 0.028168028974, -0.091920010569, -0.052043163181, 0.421566206733,

0.774289603730, 0.437991626216, -0.062035963969, -0.105574208714, 0.041289208754,
0.032683574270, -0.019761778945, -0.009164231163, 0.006764185449, 0.002433373213,
-0.001662863702, -0.000638131343, 0.000302259582, 0.000140541150, -0.000041340432,
-0.000021315027, 0.000003734655, 0.000002063762, -0.000000167443, -0.000000095177};

- Beylkin18 :

{0.099305765374353, 0.424215360812961, 0.699825214056600,
0.449718251149468, -0.110927598348234, -0.264497231446384, 0.026900308803690,
0.155538731877093, -0.017520746266529, -0.088543630622924, 0.019679866044322,
0.042916387274192, -0.017460408696028, -0.014365807968852, 0.010040411844631,
0.0014842347824723, -0.002736031626258, 0.0006404853285212};

- Vaidyanathan24:

{-0.000062906118, 0.000343631905, -0.000453956620, -0.000944897136, 0.002843834547,
0.000708137504, -0.008839103409, 0.003153847056, 0.019687215010, -0.014853448005,
-0.035470398607, 0.038742619293, 0.055892523691, -0.077709750902, -0.083928884366,
0.131971661417, 0.135084227129, -0.194450471766, -0.263494802488, 0.201612161775,
0.635601059872, 0.572797793211, 0.250184129505, 0.045799334111}.

Apêndice II – Código fonte do algoritmo.

```
#include<iostream.h>
#include<stdio.h>
#include<math.h>
#include<string.h>
#include "wavelet.h"
//-----
main(int i,char* n[])
{
    short converte2de8para1de16(unsigned char, unsigned char);
    void segment(double*,long,unsigned long,int);

    int t=atoi(n[2]);

    FILE* fr;

    if(((fr=fopen(n[1],"rb"))!=NULL))
    {
        struct
        {
            unsigned char riff[4];
            unsigned long len;
        } riff_header;
        fread(&riff_header,sizeof(riff_header),1,fr);
        cout<<"\nArquivo do tipo:
"<<<riff_header.riff[0]<<<riff_header.riff[1]<<<riff_header.riff[2]<<<riff_header.riff[3];
        cout<<"\nTamanho excluindo header: "<<<riff_header.len;
        ///////////////////////////////////
        unsigned char wave[4];
        fread(&wave,sizeof(wave),1,fr); //
        cout<<"\nSub-Tipo: "<<<wave[0]<<<wave[1]<<<wave[2]<<<wave[3];
        ///////////////////////////////////
        struct
        {
            unsigned char id[4];
            unsigned long len;
        } riff_chunk;
        fread(&riff_chunk,sizeof(riff_chunk),1,fr);
        cout<<"\nIdentificador: "<<<riff_chunk.id[0]<<<riff_chunk.id[1]<<<riff_chunk.id[2]<<<riff_chunk.id[3];
        cout<<"\nComprimento do chunk apos header: "<<<riff_chunk.len;

        ///////////////////////////////////
        struct
        {
            unsigned short formattag;
            unsigned short numberofchannels;
            unsigned long samplingrate;
            unsigned long avgbytespersecond;
            unsigned short blockalign;
```

```

    } wave_chunk;
    fread(&wave_chunk,sizeof(wave_chunk),1,fr);
    //tratamento de uma excessao que costuma aparecer em alguns arquivos wav... O correto seriam 16 bytes,
    as vezes aparecem 18 ou mais...
    if(riff_chunk.len>16)
    {
        unsigned char excesso;
        for(int i=0;i<riff_chunk.len-16;i++)
        {
            fread(&excesso,sizeof(excesso),1,fr);
        }
    }
    //fim do tratamento da excess?
    cout<<"\nCategoria do formato: "<<wave_chunk.formattag;
    cout<<"\nNumero de canais: "<<wave_chunk.numberofchannels;
    cout<<"\nTaxa de amostragem: "<<wave_chunk.samplingrate;
    cout<<"\nMedia do num. de bps: "<<wave_chunk.avgbytespersecond;
    cout<<"\nAlinhamento do bloco em bytes: "<<wave_chunk.blockalign;
    //////////////////////////////////////

    if(wave_chunk.formattag==1) //PCM
    {
        int resolucao=(wave_chunk.avgbytespersecond * 8)/(wave_chunk.numberofchannels *
        wave_chunk.samplingrate);// pq nao bitssample
        cout<<"\nResolucao: "<<resolucao;

        struct
        {
            unsigned char data[4];
            unsigned long chunk_size;
        } header_data_chunk;

        fread(&header_data_chunk,sizeof(header_data_chunk),1,fr);
        cout<<"\nIdentificacao:
        "<<header_data_chunk.data[0]<<header_data_chunk.data[1]<<header_data_chunk.data[2]<<header_data_c
        hunk.data[3];
        cout<<"\nTamanho do chunk de dados: "<<header_data_chunk.chunk_size;
        cout<<"\nNumero de frames para amostrar:
        "<<header_data_chunk.chunk_size/wave_chunk.blockalign;

        long comprimento_do_sinal=(header_data_chunk.chunk_size/wave_chunk.blockalign);
        cout<<"\nComprimento do sinal: "<<comprimento_do_sinal;

        if((resolucao==8) && (wave_chunk.numberofchannels==1))
        {
            unsigned char waveformdata;
            double* amostras_no_f = new
            double[(header_data_chunk.chunk_size/wave_chunk.blockalign)];
            for(long i=0;i<(header_data_chunk.chunk_size/wave_chunk.blockalign);i++)
            {
                fread(&waveformdata,sizeof(waveformdata),1,fr);
                amostras_no_f[i]=(double)waveformdata;
            }
            segment(&amostras_no_f[0],comprimento_do_sinal,wave_chunk.samplingrate,t);
        }
        else if((resolucao==8) && (wave_chunk.numberofchannels==2))
        {
            unsigned char waveformdata_right;    109
            unsigned char waveformdata_left;
            double* amostras_no_f_left = new
            double[(header_data_chunk.chunk_size/wave_chunk.blockalign)];
            double* amostras_no_f_right = new
            double[(header_data_chunk.chunk_size/wave_chunk.blockalign)];

```

```

        for(long i=0;i<(header_data_chunk.chunk_size/wave_chunk.blockalign);i++)
        {
            fread(&waveformdata_left,sizeof(waveformdata_left),1,fr);
            fread(&waveformdata_right,sizeof(waveformdata_right),1,fr);
            amostras_no_f_right[i]=(double)waveformdata_right;
            amostras_no_f_left[i]=(double)waveformdata_left;
        }
        segment(&amostras_no_f_right[0],comprimento_do_sinal,wave_chunk.samplingrate,t);
        segment(&amostras_no_f_left[0],comprimento_do_sinal,wave_chunk.samplingrate,t);
    }
    else if((resolucao==16) && (wave_chunk.numberofchannels==1))
    {
        unsigned char waveformdata_lsb, waveformdata_msb;
        double* amostras_no_f = new
double[(header_data_chunk.chunk_size/wave_chunk.blockalign)];
        for(long i=0;i<(header_data_chunk.chunk_size/wave_chunk.blockalign);i++)
        {
            fread(&waveformdata_lsb,sizeof(waveformdata_lsb),1,fr);
            fread(&waveformdata_msb,sizeof(waveformdata_msb),1,fr);
            amostras_no_f[i]=(double)converte2de8para1de16(waveformdata_lsb,waveformdata_ms
b);
        }
        segment(&amostras_no_f[0],comprimento_do_sinal,wave_chunk.samplingrate,t);
    }
    else if ((resolucao==16) && (wave_chunk.numberofchannels==2))
    {
        unsigned char waveformdata_lsb_left, waveformdata_lsb_right, waveformdata_msb_left,
waveformdata_msb_right;
        double* amostras_no_f_left = new
double[(header_data_chunk.chunk_size/wave_chunk.blockalign)];
        double* amostras_no_f_right = new
double[(header_data_chunk.chunk_size/wave_chunk.blockalign)];
        for(long i=0;i<(header_data_chunk.chunk_size/wave_chunk.blockalign);i++)
        {
            fread(&waveformdata_lsb_left,sizeof(waveformdata_lsb_left),1,fr);
            fread(&waveformdata_msb_left,sizeof(waveformdata_msb_left),1,fr);
            fread(&waveformdata_lsb_right,sizeof(waveformdata_lsb_right),1,fr);
            fread(&waveformdata_msb_right,sizeof(waveformdata_msb_right),1,fr);
            amostras_no_f_left[i]=(double)converte2de8para1de16(waveformdata_lsb_left,wavefor
mdata_msb_left);
            amostras_no_f_right[i]=(double)converte2de8para1de16(waveformdata_lsb_right,wavef
ormdata_msb_right);
        }
        segment(&amostras_no_f_right[0],comprimento_do_sinal,wave_chunk.samplingrate,t);
        segment(&amostras_no_f_left[0],comprimento_do_sinal,wave_chunk.samplingrate,t);
    }
    else
    {
        cout<<"Resolucao ou nmero de canais invalido(s)";
        exit(0);
    }
}
else
    cout<<"FORA DO FORMATO PCM...";
fclose(fr);
}
else
    cout<<"Arquivo nao existe ou nao pode ser aberto";
cout<<"\n\n";
}
//-----
short converte2de8para1de16(unsigned char lsb, unsigned char msb)
{

```

```

return(((msb&0x80)>>7)*(32768) +
      ((msb&0x40)>>6)*(16384) +
      ((msb&0x20)>>5)*(8192) +
      ((msb&0x10)>>4)*(4096) +
      ((msb&0x08)>>3)*(2048) +
      ((msb&0x04)>>2)*(1024) +
      ((msb&0x02)>>1)*(512) +
      (msb&0x01))*(256) +
      ((lsb&0x80)>>7)*(128) +
      ((lsb&0x40)>>6)*(64) +
      ((lsb&0x20)>>5)*(32) +
      ((lsb&0x10)>>4)*(16) +
      ((lsb&0x08)>>3)*(8) +
      ((lsb&0x04)>>2)*(4) +
      ((lsb&0x02)>>1)*(2) +
      (lsb&0x01));
}
//-----
void segment(double* sinal, long comprimento_do_sinal, unsigned long taxa_de_amostragem, int
tamanho_da_janela)
{
void filtro_fir_passa_baixa_fase_linear(double*, long, int, double);
double fractal_dimension(double*,long);
double energia(double*,long);

FILE* arquivo_texto_dos_pontos_originais;
arquivo_texto_dos_pontos_originais=fopen("original_signal.data","w");
for(long i=0;i<comprimento_do_sinal;i++)
    fprintf(arquivo_texto_dos_pontos_originais,"%d\t%.3f\n",i,sinal[i]);
fclose(arquivo_texto_dos_pontos_originais);
double* s=new double[tamanho_da_janela];
double* fd=new double[comprimento_do_sinal];
double* en=new double[comprimento_do_sinal];
int k=0;
double media;
double desvio;

FILE* arquivo_texto_dos_pontos_transformados;
arquivo_texto_dos_pontos_transformados=fopen("fractal_dimensions.data","w");
for(long i=0;i<comprimento_do_sinal;i+=(int)((double)tamanho_da_janela/2.0))
{
    for(long j=0;j<tamanho_da_janela;j++)
    {
        s[j]=sinal[i+j];
        if(s[j]==0)
            s[j]=1;
    }
    //subtrai a media
    media=0;
    for(long j=0;j<tamanho_da_janela;j++)
        media+=s[j];
    media/=(double)(tamanho_da_janela);

    //divide pelo desvio
    desvio=0;
    for(long j=0;j<tamanho_da_janela;j++)
        desvio+=s[j]*s[j];
    desvio/=(double)(tamanho_da_janela);
    desvio-=media*media;
    for(long j=0;j<tamanho_da_janela;j++)
        s[j]-=media;
    for(long j=0;j<tamanho_da_janela;j++)
        s[j]/=desvio;
}

```



```

        en[k]=energia(&s[0],tamanho_da_janela);
        fd[k]=fractal_dimension(&s[0],tamanho_da_janela);
//    printf(" \t\t FD = %.3f",fd[k]);
        fprintf(arquivo_texto_dos_pontos_transformados,"%d\t%.3f\n",k,fd[k]);
        k++;
    }
    fclose(arquivo_texto_dos_pontos_transformados);

    int sfd=k-1;
    for(long j=0;j<sfd;j++)
        fd[j]=((fd[j+1]-fd[j])>0)?(fd[j+1]-fd[j]):(fd[j]-fd[j+1]);
    filtro_fir_passa_baixa_fase_linear(&fd[0], sfd, 100, 0.01);
    FILE* arquivo_texto_dos_pontos_transformados_derivados;
    arquivo_texto_dos_pontos_transformados_derivados=fopen("derivatives_fractal_dimensions.data","w");
    for(long i=0;i<sfd;i++)
    {
        fprintf(arquivo_texto_dos_pontos_transformados_derivados,"%d\t%.3f\t%.3f\n",i,fd[i]);
        printf("\nINICIO = %d  COMPR = %d  DER_FD = %.3f  ENE = %.3f",
            (int)(i*tamanho_da_janela/2.0),tamanho_da_janela,fd[i],en[i]);
    }
    fclose(arquivo_texto_dos_pontos_transformados_derivados);
    cout<<"\n\n\n";
    for(long i=1;i<sfd-1;i++)
        if(((fd[i]<fd[i-1]]&&(fd[i]<fd[i+1]))||((fd[i]>fd[i-1]]&&(fd[i]>fd[i+1]))))
            printf("\nTRANS. = %d", (int)(i*tamanho_da_janela/2.0));
    }
//-----
double energia(double* sinal, long comprimento_do_sinal)
{
    double e=0;
    for (long i=0;i<comprimento_do_sinal;i++)
        e+=sinal[i]*sinal[i];
    return(e);
}
//-----
double fractal_dimension(double* sinal, long comprimento_do_sinal)
{
    double h[]={4.53853736157e-03, 4.67433948927e-02, 2.06023863986e-01, 4.92631771708e-01,
        6.45813140357e-01, 3.39002535454e-01,-1.93204139609e-01,-2.88882596566e-01, 6.52829528487e-02,
        1.90146714007e-01,-3.96661765557e-02,-1.11120936037e-01, 3.38771439235e-02, 5.47805505845e-02,-
        2.57670073284e-02,-2.08100501696e-02, 1.50839180278e-02, 5.10100036040e-03,-6.48773456031e-03,-
        2.41756490761e-04, 1.94332398038e-03,-3.73482354137e-04,-3.59565244362e-04, 1.55896489920e-04,
        2.57926991553e-05,-2.81332962660e-05, 3.36298718173e-06, 1.81127040794e-06,-6.31688232588e-07,
        6.13335991330e-08};
    int ch=sizeof(h)/sizeof(double); //ch, that is automatically determined, is the size of h[]

    //decomposicao no nivel maximo possivel
    transformada_wavelet_nivel_k(&sinal[0],comprimento_do_sinal,(int)(log(comprimento_do_sinal)/log(2)), 'n',
        h,ch);

    double* energia_da_banda=new double[(int)(log(comprimento_do_sinal)/log(2))]; //todos os detalhes (1 de
    //cada nivel) desconsiderando a aproximacao do ultimo nivel
    double* comprimento_da_banda=new double[(int)(log(comprimento_do_sinal)/log(2))]; //comprimento:
    //todos os detalhes (1 de cada nivel) desconsiderando a aproximacao do ultimo nivel
    int inicio=1;
    int comprimento=1;
    for(int i=0;i<(int)(log(comprimento_do_sinal)/log(2));i++)
    {
        energia_da_banda[i]=0;
        for(int j=inicio;j<inicio+comprimento;j++)
            energia_da_banda[i]+=sinal[j]*sinal[j];
        energia_da_banda[i]/=(double)(comprimento);
        comprimento_da_banda[i]=comprimento;
        inicio*=2;
    }
}

```

```

        comprimento*=2;
    }

    double sy=0;
    double sx=0;
    double sxy=0;
    double sx2=0;

    for(long i=0;i<(int)(log(comprimento_do_sinal)/log(2));i++)
    {
        sx+=log2(comprimento_da_banda[i]);
        sx2+=log2(comprimento_da_banda[i])*log2(comprimento_da_banda[i]);
        sy+=log2(energia_da_banda[i]);
        sxy+=log2(comprimento_da_banda[i])*log2(energia_da_banda[i]);
    }

    double b=((int)(log(comprimento_do_sinal)/log(2))*sxy-
    sy*sx)/((int)(log(comprimento_do_sinal)/log(2))*sx2-sx*sx);
    double fd=(b-1.0)/2.0;
    if(fd<0)
        fd*=-1.0;
    fd=2-fd;
    //resposta
    return(fd);
}
//-----
void filtro_fir_passa_baixa_fase_linear(double* u, long comprimento, int m, double wc) //m e'a ordem e wc
a freq de corte angular. O valor max de wc e'l
{
    double c[m+1];
    for(int n=0;n<m+1;n++)
    {
        if(((double)n-(double)m/2.0)!=0)
            c[n]=((sin(wc*3.1415927*((double)(n)-(double)(m)/2.0)))/(3.1415927*((double)(n)-
            (double)(m)/2.0))) * (0.42-
            0.5*cos(2.0*3.1415927*(double)(n)/(double)(m))+0.08*cos(4.0*3.1415927*(double)(n)/(double)(m)));
        else
            c[n]=0;
    }
    double s=0;
    for(int i=0;i<(int)(sizeof(c)/sizeof(double));i++)
        s+=c[i];
    for(int i=0;i<(int)(sizeof(c)/sizeof(double));i++)
        c[i]*=1.0/(s*s);

    double* y=new double[comprimento+(int)(sizeof(c)/sizeof(double))-1];

    for(long n=0;n<comprimento+(int)(sizeof(c)/sizeof(double))-1;n++)
    {
        y[n]=0;
        for(int k=0;k<(int)(sizeof(c)/sizeof(double));k++)
            if((n-k)>=0)
                y[n]+=u[n-k]*c[k];
    }
    for(long i=0;i<comprimento;i++)
        u[i]=y[i];
}
//wavelet.h
//bibliotecas de funcoes de transformada wavelet

```

const long qtd_max_col_matriz=4; //quantidade de colunas da matriz bidimensioanl, usada apenas
//nas funcoes bidimensionais. Modificar isso, conforme a quantidade de colunas da matriz a ser
//usada no arquivo cpp. O valor 4 acima e' apenas um exemplo.

```

//-----
void transformada_wavelet_nivel_k(double* f, long n, int nivel, char ordem, double h[], int ch)
{
    //double h[] = {1/sqrt(2), 1/sqrt(2)};
    //int ch = sizeof(h)/sizeof(double);
    double *g = new double[ch];
    for(int i=0; i<ch; i++)
    {
        g[i] = h[ch-i-1];
        if(i%2!=0)
            g[i] *= -1;
    }
    int cg = ch;
    long j = 0;
    double* t = new double[n];
    if(ordem=='n') // n de normal para wavelet
    {
        for(long i=0; i<n; i+=2) //trend
        {
            t[j] = 0;
            for(long k=0; k<ch; k++)
                t[j] += f[(i+k)%n] * h[k];
            j++;
        }
        for(long i=0; i<n; i+=2) //fluctuation
        {
            t[j] = 0;
            for(long k=0; k<cg; k++)
                t[j] += f[(i+k)%n] * g[k];
            j++;
        }
    }
    else // i de invertido para wavelet packet
    {
        for(long i=0; i<n; i+=2) //fluctuation
        {
            t[j] = 0;
            for(long k=0; k<cg; k++)
                t[j] += f[(i+k)%n] * g[k];
            j++;
        }
        for(long i=0; i<n; i+=2) //trend
        {
            t[j] = 0;
            for(long k=0; k<ch; k++)
                t[j] += f[(i+k)%n] * h[k];
            j++;
        }
    }
    for(long i=0; i<n; i++)
        f[i] = t[i];
    nivel--;
    n /= 2;
    delete(t);
    if(nivel>0)
        transformada_wavelet_nivel_k(&t[0], n, nivel, ordem, h, ch);
}
//-----
void transformada_wavelet_packet_nivel_k(double* f, long n, int nivel, double h[], int ch)
{
    long inicio = 0;
    long comprimento = n;
    for(int i=1; i<=nivel; i++) // por exemplo, para nivel 5, vou chamar 5 vezes a fun?o de transformada, cada

```

vez em n?el 1.

```

{
    inicio=0;
    comprimento=(int)(n/pow(2,i-1));
    for(int j=0;j<pow(2,i-1);j++)
    {
        if(j%2==0)
            transformada_wavelet_nivel_k(&f[inicio],comprimento,1,'n',h,ch); // n de ordem normal:
            primeiro passa-baixa e depois passa-alta
        else
            transformada_wavelet_nivel_k(&f[inicio],comprimento,1,'i',h,ch); // i de invertido: primeiro
            passa-alta e depois passa-baixa
        inicio+=comprimento;
    }
}
}
//-----

```

```

void transformada_wavelet_inversa_nivel_k(double* f, long n, int nivel, char ordem, double h[], int ch)
{

```

```

//double h[]={1/sqrt(2),1/sqrt(2)};

```

```

//int ch=sizeof(h)/sizeof(double);

```

```

double* g=new double[ch];

```

```

for(int i=0;i<ch;i++)
{

```

```

    g[i]=h[ch-i-1];

```

```

    if(i%2!=0)

```

```

        g[i]*=-1;
    }

```

```

double* sfi=new double[ch];

```

```

for(int i=0;i<ch;i+=2)
    sfi[i]=h[ch-2-i];

```

```

for(int i=0;i<ch;i+=2)
    sfi[i+1]=g[ch-2-i];

```

```

int csfi=ch;

```

```

double *wfi=new double[csfi];

```

```

for(int i=0;i<ch;i+=2)
    wfi[i]=h[ch-1-i];

```

```

for(int i=0;i<ch;i+=2)
    wfi[i+1]=g[ch-1-i];

```

```

long comprimento_do_subsinal=2*(long)((n)/(pow(2,nivel)));

```

```

double* subsinal=new double[comprimento_do_subsinal];

```

```

if(ordem=='n') //normal
{

```

```

    for(long i=0;i<comprimento_do_subsinal;i+=2)
    {

```

```

        subsinal[i]=f[(int)(i/2)];

```

```

        subsinal[i+1]=f[(int)(i/2)+(int)(comprimento_do_subsinal/2)];
    }

```

```

}

```

```

else // i // invertido -> alguns casos da packet
{

```

```

    for(long i=0;i<comprimento_do_subsinal;i+=2)
    {

```

```

        subsinal[i]=f[(int)(i/2)+(int)(comprimento_do_subsinal/2)];

```

```

        subsinal[i+1]=f[(int)(i/2)];
    }

```

```

}

```

```

long start;

```

```

if(comprimento_do_subsinal>=csfi)

```

```

    start=abs(comprimento_do_subsinal-csfi)+2;

```

```

else

```

```

    {
        long comprimento_matricial_do_sinal=2;
        while(comprimento_matricial_do_sinal<csfi)
            comprimento_matricial_do_sinal+=comprimento_matricial_do_sinal;
        start=comprimento_matricial_do_sinal-csfi+2;
    }

for(long j=0;j<comprimento_do_subsinal;j+=2)
{
    f[j]=0;
    f[j+1]=0;
    for(int k=0;k<csfi;k++)
    {
        f[j]+=sfi[k]*subsinal[(start+k)%(comprimento_do_subsinal)];
        f[j+1]+=wfi[k]*subsinal[(start+k)%(comprimento_do_subsinal)];
    }
    start+=2;
}
nivel--;
delete(subsinal);
if(nivel>0)
    transformada_wavelet_inversa_nivel_k(&f[0],n,nivel,ordem,h,ch);
}
//-----
void transformada_wavelet_packet_inversa_nivel_k(double* f, long n, int nivel, double h[], int ch)
{
    long inicio=0;
    long comprimento=n;
    for(int i=nivel;i>=1;i--) // por exemplo, para nivel 5, vou chamar 5 vezes a fun?o de tranasformada, cada
        vez em n?el 1.
    {
        inicio=0;
        comprimento=(int)(n/pow(2,i-1));
        for(int j=0;j<pow(2,i-1);j++)
        {
            if(j%2==0)
                transformada_wavelet_inversa_nivel_k(&f[inicio],comprimento,1,'n',h,ch); // n de ordem
                normal: primeiro passa-baixa e depois passa-alta
            else
                transformada_wavelet_inversa_nivel_k(&f[inicio],comprimento,1,'i',h,ch); // i de invertido:
                primeiro passa-alta e depois passa-baixa
                inicio+=comprimento;
        }
    }
}
//-----
void transformada_wavelet_bidimensional_nivel_k(double f[][qtd_max_col_matriz], long li, long ci, long n,
long m, int nivel, char ordem_l, char ordem_c, double h[], int ch)
{
    double* vetor_linha=new double[m];
    for(int i=li;i<li+m;i++)
    {
        for(int j=ci;j<ci+m;j++)
            vetor_linha[j-ci]=f[i][j];
        transformada_wavelet_nivel_k(&vetor_linha[0],m,1,ordem_l,h,ch);
        for(int j=ci;j<ci+m;j++)
            f[i][j]=vetor_linha[j-ci];
    }
    double* vetor_coluna=new double[n];
    for(int j=ci;j<ci+m;j++)
    {
        for(int i=li;i<li+n;i++)
            vetor_coluna[i-li]=f[i][j];
        transformada_wavelet_nivel_k(&vetor_coluna[0],n,1,ordem_c,h,ch);
    }
}

```

```

        for(int i=li;i<li+n;i++)
            f[i][j]=vetor_coluna[i-li];
    }
    nivel--;
    n/=2;
    m/=2;
    delete(vetor_linha);
    delete(vetor_coluna);
    if(nivel>0)
        transformada_wavelet_bidimensional_nivel_k(f, li, ci, n, m, nivel, ordem_l, ordem_c,h,ch);
}
//-----
void transformada_wavelet_bidimensional_inversa_nivel_k(double f[][qtd_max_col_matriz], long li, long
ci, long n, long m, int nivel, char ordem_l, char ordem_c, double h[], int ch)
{
    long quantidade_de_linhas_da_submatriz=2*(long)((n)/(pow(2,nivel)));
    long quantidade_de_colunas_da_submatriz=2*(long)((m)/(pow(2,nivel)));

    double* vetor_linha=new double[quantidade_de_colunas_da_submatriz];
    for(int i=li;i<li+quantidade_de_linhas_da_submatriz;i++)
    {
        for(int j=ci;j<ci+quantidade_de_colunas_da_submatriz;j++)
            vetor_linha[j-ci]=f[i][j];
        transformada_wavelet_inversa_nivel_k(&vetor_linha[0],quantidade_de_colunas_da_submatriz,1,orde
m_l,h,ch);
        for(int j=ci;j<ci+quantidade_de_colunas_da_submatriz;j++)
            f[i][j]=vetor_linha[j-ci];
    }
    double* vetor_coluna=new double[quantidade_de_linhas_da_submatriz];
    for(int j=ci;j<ci+quantidade_de_colunas_da_submatriz;j++)
    {
        for(int i=li;i<li+quantidade_de_linhas_da_submatriz;i++)
            vetor_coluna[i-li]=f[i][j];
        transformada_wavelet_inversa_nivel_k(&vetor_coluna[0],quantidade_de_linhas_da_submatriz,1,orde
m_c,h,ch);
        for(int i=li;i<li+quantidade_de_linhas_da_submatriz;i++)
            f[i][j]=vetor_coluna[i-li];
    }
    nivel--;
    delete(vetor_linha);
    delete(vetor_coluna);
    if(nivel>0)
        transformada_wavelet_bidimensional_inversa_nivel_k(f, li, ci, n, m, nivel, ordem_l, ordem_c,h,ch);
}
//-----
void transformada_wavelet_packet_bidimensional_nivel_k(double f[][qtd_max_col_matriz], long n, long m,
int nivel, double h[], int ch)
{
    long inicio_da_linha;
    long inicio_da_coluna;
    long comprimento_da_linha;
    long comprimento_da_coluna;

    for(int k=1;k<=nivel;k++) // por exemplo, para nivel 5, vou chamar 5 vezes a fun?o de transformada, cada
vez em n?el 1.
    {
        inicio_da_linha=0;
        inicio_da_coluna=0;
        comprimento_da_linha=(int)(m/pow(2,k-1));
        comprimento_da_coluna=(int)(n/pow(2,k-1));
        for(int i=0;i<pow(2,k-1);i++)
        {

```

```

        inicio_da_coluna=0;
        for(int j=0;j<pow(2,k-1);j++)
        {
            if((i%2==0)&&(j%2==0))
                transformada_wavelet_bidimensional_nivel_k(f,inicio_da_linha, inicio_da_coluna,
comprimento_da_linha,comprimento_da_coluna,1,'n','n',h,ch);
            else if ((i%2==0)&&(j%2!=0))
                transformada_wavelet_bidimensional_nivel_k(f,inicio_da_linha, inicio_da_coluna,
comprimento_da_linha,comprimento_da_coluna,1,'T','n',h,ch);
            else if ((i%2!=0)&&(j%2==0))
                transformada_wavelet_bidimensional_nivel_k(f,inicio_da_linha, inicio_da_coluna,
comprimento_da_linha,comprimento_da_coluna,1,'n','T',h,ch);
            else
                transformada_wavelet_bidimensional_nivel_k(f,inicio_da_linha, inicio_da_coluna,
comprimento_da_linha,comprimento_da_coluna,1,'T','T',h,ch);
            inicio_da_coluna+=comprimento_da_coluna;
        }
        inicio_da_linha+=comprimento_da_linha;
    }
}
//-----
void transformada_wavelet_packet_bidimensional_inversa_nivel_k(double f[][qtd_max_col_matriz], long n,
long m, int nivel, double h[], int ch)
{
    long inicio_da_linha;
    long inicio_da_coluna;
    long comprimento_da_linha;
    long comprimento_da_coluna;

    for(int k=nivel;k>=1;k--)
    {
        inicio_da_linha=0;
        inicio_da_coluna=0;

        comprimento_da_linha=(int)(m/pow(2,k-1));
        comprimento_da_coluna=(int)(n/pow(2,k-1));

        for(int i=0;i<pow(2,k-1);i++)
        {
            inicio_da_coluna=0;
            for(int j=0;j<pow(2,k-1);j++)
            {
                if((i%2==0)&&(j%2==0))
                    transformada_wavelet_bidimensional_inversa_nivel_k(f,inicio_da_linha,
inicio_da_coluna, comprimento_da_linha,comprimento_da_coluna,1,'n','n',h,ch);
                else if ((i%2==0)&&(j%2!=0))
                    transformada_wavelet_bidimensional_inversa_nivel_k(f,inicio_da_linha,
inicio_da_coluna, comprimento_da_linha,comprimento_da_coluna,1,'T','n',h,ch);
                else if ((i%2!=0)&&(j%2==0))
                    transformada_wavelet_bidimensional_inversa_nivel_k(f,inicio_da_linha,
inicio_da_coluna, comprimento_da_linha,comprimento_da_coluna,1,'n','T',h,ch);
                else
                    transformada_wavelet_bidimensional_inversa_nivel_k(f,inicio_da_linha,
inicio_da_coluna, comprimento_da_linha,comprimento_da_coluna,1,'T','T',h,ch);
                inicio_da_coluna+=comprimento_da_coluna;
            }
            inicio_da_linha+=comprimento_da_linha;
        }
    }
}
//-----

```

Apêndice III - Publicações durante o mestrado.

Durante o curso de mestrado do autor, os seguintes artigos foram publicados:

- FANTINATO, P. C. et al. *A Fractal-based Approach for Speech Segmentation*. In: 10th IEEE International Symposium on Multimedia (ISM2008): Berkeley, CA, USA. p. in–press, 2008.
- GUIDO, R. C.; MACIEL, C. D.; MONTEIRO, M., FONSECA, E., PANCHAPAGESAN, S.; PEREIRA, J. C.; VIEIRA, L.; BARBON, S.Jr.; SANCHEZ, F.L.; GUILHERME, M.B.A.; SERGIO, K.I.C.; SCARPA, T.L.; FANTINATO, P.C.; MOURA, E.J.R. *A study on the best wavelet for audio compression*. In: 40th IEEE ASILOMAR Int. Conference on Signals, Systems and Computers, Pacific-Grove, CA, USA, 2006, p.2115-2118.
- GUIDO, R. C., BARBON JR, S., VIEIRA, L. S., RODRIGUES, L. C., FANTINATO, P. C., ESCOLA, J. P. L., SOUZA, L. M., ALMEIDA, L. O. B., SLAETS, J. F. W. *Autenticação biométrica por comando de voz com implementação em hardware dedicado*. In: II Workshop no âmbito do projecto - Ambientes para co-projeto hardware/software em plataformas FPGAs com aplicação em robótica móvel, São Carlos, 2008 (CD-ROM).
- GUIDO, R.C., JUNIOR, S.B., VIEIRA, L.S., FANTINATO, P. C., SANCHEZ, F.L., ESCOLA, J.P.; BOLOGNINI, E., SOUZA, L., ZULATO, P., RIBEIRO, J., LACERDA, M. *Speech Compression Based on the Discrete Shapelet Transform*. In: XXXI CNMAC - Congresso Nacional de Matemática Aplicada e Computacional, Recife-PE, 2008 (CR-ROM).

- VIEIRA, L.S., GUIDO, R.C., BARBON Jr, S., SANCHEZ, F.L., SERGIO, K.I.C., GUILHERME, M.B.A., SOUZA, L.M., FANTINATO, P. C., BASSI, R.D.S., AMADIO, B.C.G., FONSECA, E.S. *On the use of wavelets for voice morphing*. In: XXX CNMAC - Congresso Nacional de Matemática Aplicada e Computacional, Florianópolis-SC, 2007 (CD-ROM).

Referências Bibliográficas

- [1] BARBON Jr, S; GUIDO, R.C. *Dynamic Time Warping baseado na Transformada Wavelet*. 2007. 113p. Dissertação de Mestrado. Instituto de Física de São Carlos da Universidade de São Paulo, 2007.
- [2] VIEIRA, L.S.; GUIDO, R.C. *Conversão de voz baseada na Transformada Wavelet*. 2007. 91p. Dissertação de Mestrado. Instituto de Física de São Carlos da Universidade de São Paulo, 2007.
- [3] HAYKIN, S.; VEEN, B.V. *Sinais e sistemas*. Porto Alegre: Bookman, 2002.
- [4] FANTINATO, P. C. et al. *A Fractal-based approach for speech segmentation*. In: 10th IEEE International Symposium on Multimedia (ISM2008), Berkeley, CA, USA, p. in-press, 2008.
- [5] ERIK, L.J.; BOHEZ, T.R. Speech recognition using fractals. *Pattern Recognition*, v. 34, p. 2227 - 2243, 2007.
- [6] AL-AKAIDI, M.; BLACKLEDGE, J.; FEKKAI, S. *Fractal dimension segmentation: isolated speech recognition*. IEE Seminar on Speech Coding for Algorithms for Radio Channels, London, UK: Fac. of Comput. Sci. and Eng., De Montfort Univ., Leicester, p.4/1 - 4/5, 2000.

- [7] MYERS, L. et al. *Performance tradeoffs in Dynamic Time Warping algorithms for isolated word recognition*. IEEE Transactions On Acoustics, Speech, And Signal Processing, v. assp-28, n. 6, 1980.
- [8] BARBON JR, S. et al. Wavelet-based Dynamic Time Warping. *Journal of Computational and Applied Mathematics*, Elsevier, v. -, p. in–press, 2007.
- [9] Base de Dados TIMIT do *Linguistic Data Consortium*, <http://www.ldc.upenn.edu/>
- [10] BOSI, M.; GOLDBERG, R.E. *Introduction to digital audio coding and standards*. Boston-USA: Kluwer, 2003.
- [11] AL-AKAIDI, M. *Fractal speech processing*. Cambridge-UK: Cambridge University Press, 2004.
- [12] SANCHEZ, F.L.; MACIEL, C.D. *Análise cepstral baseada em diferentes famílias de transformadas wavelet*. 2008. 96p. Dissertação de Mestrado. Interunidades em Bioengenharia. Escola de Engenharia de São Carlos da Universidade de São Paulo, 2008.
- [13] PICKOVER, C.A., KHORASANI, A. Fractal characterization of speech waveform graphs. *Computer and Graphics*, v.10, n.1, pp.51-61, 1986.
- [14] ADDISON, P.S. *The illustrated Wavelet Transform handbook: Introductory Theory and Applications in Science, Engineering, Medicine and Finance*. Edinburg-UK: Institute of Physics Publishing, 2002.
- [15] JENSEN, A. et al. *Ripples in mathematics: the discrete Wavelet Transform*. New York-USA: Springer-Verlag, 2000.

- [16] AKAY, M. *Time-frequency and Wavelets in biomedical signal processing*. New York: IEEE Press in Biomedical Engineering, 1998.
- [17] WILLIAMS, J.R.; AMARATUNGA, K. *Introduction to wavelets in engineering*. International Journal for Numerical Methods in Engineering, v. 37, 1994, pp. 2365-2388.
- [18] LIPSCHUTZ, S. *Álgebra linear*. 3.ed. São Paulo: Makron Books, 1994.
- [19] STRANG, G., NGUYEN, T. *Wavelets and filter-banks*. Wellesley-Cambridge Academic Press, Wellesley, 1997.
- [20] WALKER, J.S. *A primer on Wavelets and their scientific applications*. Whashington-USA: Chapman and Hall/CRC, 1999.
- [21] BURIAN, R.; LIMA, A.C. *Cálculo numérico*, 1.ed., Rio de Janeiro: LTC, 2007.
- [22] SCHILDT, H. *C++: the complete reference*, 4.ed., McGraw-Hill Osborne Media, 2002.
- [23] GUIDO, R.C. *Spikelet: uma nova transformada wavelet aplicada ao reconhecimento digital de padrões, em tempo-real, de spikes e overlaps em sinais neurofisiológicos do campo visual da mosca*. Tese de Doutorado. Instituto de Física de São Carlos da Universidade de São Paulo, 2003.
- [24] GUIDO, R.C. et al. *A new technique to construct a wavelet transform matching a specified signal with applications to digital, real-time, spike and overlap pattern recognition*. Digital Signal Processing, Elsevier, EUA, v. 16, n. 1, p. 24-44, 2006.