Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Programa de Pós-Graduação em Engenharia Elétrica

Reconhecimento de Fala Contínua para o Português Brasileiro em Sistemas Embarcados

Daniella Dias Cavalcante da Silva

Tese de Doutorado submetida à Coordenação dos Cursos de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, como parte dos requisitos necessários para obtenção do grau de Doutor em Ciências no domínio da Engenharia Elétrica.

Área de Concentração: Processamento da Informação

Orientador: Benedito Guimarães Aguiar Neto - Dr.-Ing.

Campina Grande, dezembro de 2011

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

S586r Silva, Daniella Dias Cavalcante da.
Reconhecimento de fala contínua para o português brasileiro em sistemas embarcados / Daniella Dias Cavalcante da Silva. – Campina Grande, 2011.
182 f. : il.

Tese (Doutorado em Engenharia Elétrica) - Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.
Orientador: Dr.-Ing. Benedito Guimarães Aguiar Neto.
Referências.

1. Reconhecimento de Fala Contínua. 2. Português Brasileiro. 3. Sistemas Embarcados. 4. Português Brasileiro. 5. Modelos Ocultos de Markov. I. Título.

CDU 621.3:004.031.6(043)

Reconhecimento de Fala Contínua para o Português Brasileiro em Sistemas Embarcados

Daniella Dias Cavalcante da Silva

Benedito Guimarães Aguiar Neto - Dr.-Ing.
Orientador

Joseana Macêdo Fechine - DSc.
Membro da Banca

Elmar Kurt Melcher - Dr.-Ing.
Membro da Banca

Francisco Marcos de Assis - Dr.-Ing.
Membro da Banca

Abraham Alcaim – PhD.
Membro da Banca

Adrião Duarte Doria Neto – Dr.Ing.
Membro da Banca

Campina Grande, Paraíba, Brasil
Dezembro/2011

Aos dois amores da minha vida,
minha mãe D. Deri e meu esposo César.

"Chegará o dia em que as máquinas pensarão,
mas nunca terão sonhos."
Theodor Heuss

"Eu penei, mas aqui cheguei."
Luíz Gonzaga

Agradecimentos

São tantas pessoas para agradecer, que qualquer ordem de citação que eu utilize, será injusta. Então, vou tentar usar uma ordem mais ou menos cronológica.

Primeiramente a Deus, pela vida, força e por mais essa grande conquista.

Aos dois amores da minha vida, D. Deri e César, pela constante confiança e incentivo, carinho, paciência e compreensão nos momentos mais difíceis dessa longa jornada.

Aos meus orientadores e verdadeiros mestres, Benedito e Joseana, pela orientação, críticas, confiança, presteza e incentivo, sempre no momento e na dose certa.

À toda minha família, especialmente minha avó Maria do Céu, meu primo e irmão Leandro, meus sogros Célia e George, meus cunhados Renata e Flávio, meu sobrinho Vítor, pelo incentivo, carinho, paciência e compreensão nas minhas muitas ausências físicas, mas nunca espirituais.

Aos amigos Georgina, Anderson, Silvana, Lúcio, Amar Ela, Karina e Maria de Lourdes pelo companheirismo, amizade e apoio.

Ao professor Elmar, pelas sugestões e esclarecimentos prestados durante o desenvolvimento deste trabalho.

A Angela, Pedro e Suênia, da Copele, sempre resolvendo todos os problemas burocráticos que surgiam.

Não foi nada fácil chegar até aqui, mas certamente teria sido impossível sem a ajuda de todos vocês.

Muito obrigada!

Resumo

Com o advento da tecnologia, as máquinas predominam em quase todos os cenários do cotidiano das pessoas, sejam essas máquinas computadores, eletrodomésticos, dispositivos portáteis, etc. Com isso, nada melhor do que dotá-las com a capacidade de percepção e compreensão da voz humana, que é a forma mais simples, natural e eficaz do ser humano expressar seus pensamentos. Apesar de muitas pesquisas na área de Processamento Digital de Sinais de Voz (PDSV) terem permitido o desenvolvimento de sistemas de Reconhecimento de Fala bastante eficientes, requisitos de processamento ainda dificultam a implementação desses sistemas em dispositivos com pequeno poder computacional, como celulares, *palm-tops* e eletrodomésticos. Para permitir a implementação de sistemas de Reconhecimento de Fala nesse contexto, alguns trabalhos sacrificam a eficiência no processo de reconhecimento em nome da redução do tamanho físico e de exigências computacionais. Assim, a busca por modelagens acústicas e linguísticas otimizadas, associadas ao uso de bases de dados representativas, pode levar a um compromisso entre desempenho do sistema em termos de taxas de reconhecimento e exigências computacionais impostas por sistemas embarcados. O objetivo principal deste trabalho consiste na modelagem da arquitetura de um sistema de reconhecimento de fala contínua para o português brasileiro, utilizando Modelos Ocultos de Markov, de forma a possibilitar sua implementação em um sistema embarcado com recursos computacionais limitados. A fim de selecionar a configuração que melhor atenda esse objetivo, foram realizados experimentos e análises, de modo a identificar possíveis adaptações, a partir de simplificações matemáticas e redução de parâmetros nas etapas do processo de reconhecimento. Em todo o trabalho, foi considerada a relação entre a taxa de reconhecimento e o custo computacional. A arquitetura do sistema embarcado desenvolvida e o seu processo de modelagem, incluindo os experimentos, as análises e os seus respectivos resultados, serão apresentados e discutidos no decorrer deste documento.

Palavras-chave: Reconhecimento de fala contínua, português brasileiro, sistemas embarcados, Modelos Ocultos de Markov.

Abstract

With the advent of technology, machines predominate in almost all scenarios of everyday life. The possibility of performing human-machine communication through speech makes this interaction easier and more productive. However, processing requirements still difficult the implementation of systems for automatic continuous speech recognition on devices with low computational power such as mobile phones, palmtops and appliances. To allow the implementation of speech recognition systems in this context, some works sacrifice efficiency in the recognition process for reducing the chip area and computational requirements. For this purpose, it becomes necessary to research for optimized acoustic and language modeling, associated with use of representative databases, looking for a good compromise between recognition rates and computational demands imposed by embedded systems. The main goal of this work is to model the architecture of a system for continuous speech recognition Brazilian Portuguese, in order to enable its implementation in an embedded system with limited computing resources. In order to select the setting that best meets this goal, experiments and analysis were performed. The purpose of these was to identify possible adaptations, from mathematical simplifications and reduction of parameters in the steps of the recognition process. During the development of this work, the relationship between recognition rate and computational cost was considered. The embedded system architecture developed and its modeling process, including experiments, analysis and their results will be presented and discussed throughout this document.

Keywords: Continuous speech recognition, brazilian portuguese, embedded system, Hidden Markov Models.

Sumário

Lista de Abreviaturas

API	*Application Programming Interface*
BPL	*Bandpass Liftering* (Filtragem passa-faixa)
CFG	*Context-Free Grammar* (Gramática Livre de Contexto)
CORDIC	*COordinate Rotation DIgital Computer*
DARPA	*Defense Advanced Research Projects Agency*
DTW	*Dynamic Time Warping* (Alinhamento Dinâmico no Tempo)
FFT	*Fast Fourier Transform* (Transformada Rápida de Fourier)
FMC	Frequência Mel Cepstral
FPGA	Field-Programmable Gate Array
GMM	*Gaussian Mixture Model*
HDL	*Hardware Description Language* (Linguagem de Descrição de *Hardware*)
HMM	*Hidden Markov Models* (Modelos Ocultos de Markov)
IDCT	*Inverse Discrete Cosine Transform* (Transformada Discreta Inversa do Cosseno)
IP Core	*Intellectual Property Core*
LPC	*Linear Predictive Coding* (Codificação por Predição Linear)
MFCC	*Mel-Frequency Cepstral Coefficients* (Coeficientes Mel Cepstrais)
NoC	*Network-on-Chip*
PB	Português Brasileiro
PDSV	Processamento Digital de Sinais de Voz
PLP	*Perceptual Linear Prediction* (Predição "Perceptual" Linear)
RAM	*Random Access Memory* (Memória de Acesso Aleatório)
ROM	*Read Only Memory* (Memória Apenas de Leitura)
RTL	*Register Transfer Level*
RTOS	*Real Time Operating System* (Sistema Operacional em Tempo Real)
SRV	Sistemas de Reconhecimento de Voz
SoC	*System on Chip* (Sistema em um Chip)
VMA	*Virtual Memory Address* - Endereço de Memória Virtual
WFST	*Weighted Finite State Transducer*

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Com o advento da tecnologia, as máquinas predominam em quase todos os cenários do cotidiano das pessoas, sejam essas máquinas computadores, eletrodomésticos, dispositivos portáteis, etc. Com isso, nada melhor do que dotá-las com a capacidade de percepção e compreensão da voz humana, que é a forma mais simples, natural e eficaz do ser humano expressar seus pensamentos. A possibilidade de realizar a comunicação homem-máquina, com a utilização da voz, torna essa interação mais fácil e produtiva [1, 7, 8, 9], uma vez que permite que as mãos e os olhos dos usuários fiquem disponíveis para outras tarefas.

Diante do crescente interesse por *softwares* e equipamentos que "compreendam", reconheçam e simulem a voz humana, pesquisas têm sido realizadas na área de Processamento Digital de Sinais de Voz (PDSV), buscando o desenvolvimento de técnicas que possibilitem a construção de padrões que permitam modelar, de forma eficiente, a fala, as características vocais únicas de cada locutor, bem como a produção da voz sintetizada [10, 11].

Sistemas de reconhecimento de voz (SRV) têm como objetivo determinar qual a palavra, a frase ou a sentença que foi pronunciada, e podem ser subdivididos em um grande número de subáreas, dependendo de alguns fatores, tais como tamanho do vocabulário e população de locutores [1, 2, 12]. Esse tipo de sistema realiza uma tarefa de reconhecimento de padrões, nesse caso padrões de fala e sempre inclui duas fases [5]: treinamento e reconhecimento (Figura 1.1). Com base nos dados de treinamento (palavras ou frases), são gerados os modelos de referência (modelo acústico), aos quais são atribuídos rótulos que identificam cada padrão (palavra ou frases). Na fase de reconhecimento, a partir dos dados de teste (sinais de voz) são obtidos padrões de teste que, em seguida, são comparados com os modelos gerados durante o treinamento e, utilizando-se uma regra de decisão, é identificado o que mais se assemelha ao padrão de entrada desconhecido [1, 13].

A etapa de pré-processamento é responsável por extrair do sinal de voz os dados relevantes e menosprezar a informação redundante, com a finalidade de repassar a informação de interesse para a etapa seguinte. Esse processo pode incluir diversas operações, dentre elas: aquisição do sinal de voz, pré-ênfase, detecção de início e fim, tratamento de ruído, separação em *frames* e janelamento [1, 14, 15].

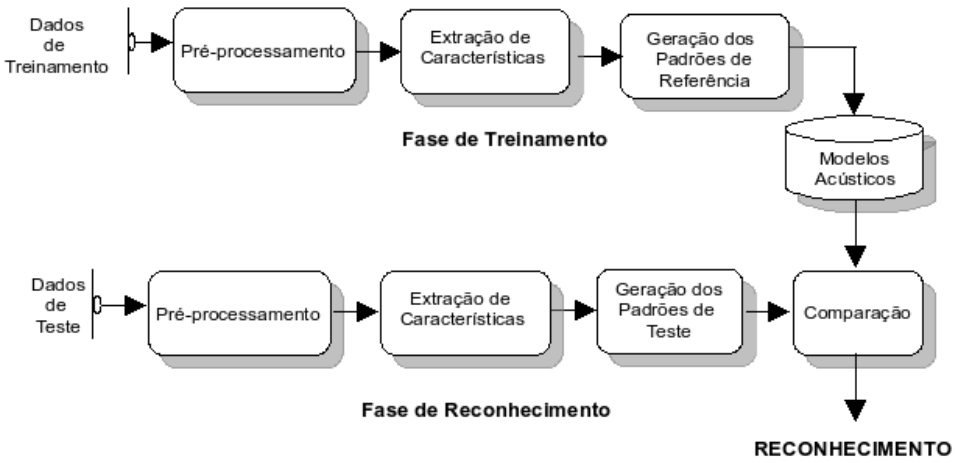


Figura 1.1: *Sistema de reconhecimento de padrões aplicado ao reconhecimento de fala (Adaptada de [1]).*

A etapa de extração de características é de extrema importância em sistemas de reconhecimento de fala, uma vez que nesta etapa são obtidos elementos que possibilitam a geração do padrão. A definição de um bom conjunto de características é um processo complexo e essencial para que se obtenha bons resultados quanto ao reconhecimento dos padrões.

Durante a fase de treinamento, com base nas características extraídas, são gerados padrões de referência, os quais serão comparados com o padrão de teste na fase de reconhecimento. Vale salientar, que as etapas de pré-processamento e extração de características das fases de treinamento e reconhecimento devem ser equivalentes, para que seja possível uma comparação correta entre o padrão testado e o(s) padrão(ões) já conhecido(s) pelo sistema.

O desenvolvimento de sistemas de reconhecimento automático da fala obteve um avanço significativo nos últimos 5 a 15 anos, em virtude, sobretudo, dos avanços também nas áreas de processamento de sinais, algoritmos, arquiteturas computacionais e *hardware* [2]. Esses avanços incluem: adoção do paradigma de reconhecimento estatístico de padrões, abordagem *data-driven*[1] (direcionada pelos dados) que faz uso de um conjunto de sentenças rico obtido de uma grande população de locutores, uso de modelagem acústica e linguística, além do uso de programação dinâmica baseada em métodos de busca [2, 16, 17].

O estado da arte dessa área pode ser direcionado de diferentes maneiras. Na Figura 1.2 é ilustrado o progresso da tecnologia de reconhecimento e entendimento da fala para aplicações em áreas genéricas, variando de palavras isoladas ou reconhecimento de comandos, até conversação natural entre homem e máquina [2, 9, 16, 18, 19, 20].

A complexidade dessas aplicações caracteriza-se ao longo de duas dimensões: o tamanho do vocabulário e a forma da elocução. Quanto maior o vocabulário, maior a dificuldade de reconhecimento. Assim como, uma conversação espontânea, incluindo expressões como "hum...", "ah!", além de palavras parciais e/ou concatenadas, são bem mais difíceis de reconhecer do que palavras pronunciadas de uma maneira estritamente articulada e pausada

[1] *Data-driven* - interpretações vão sendo realizadas à medida que os dados são analisados.

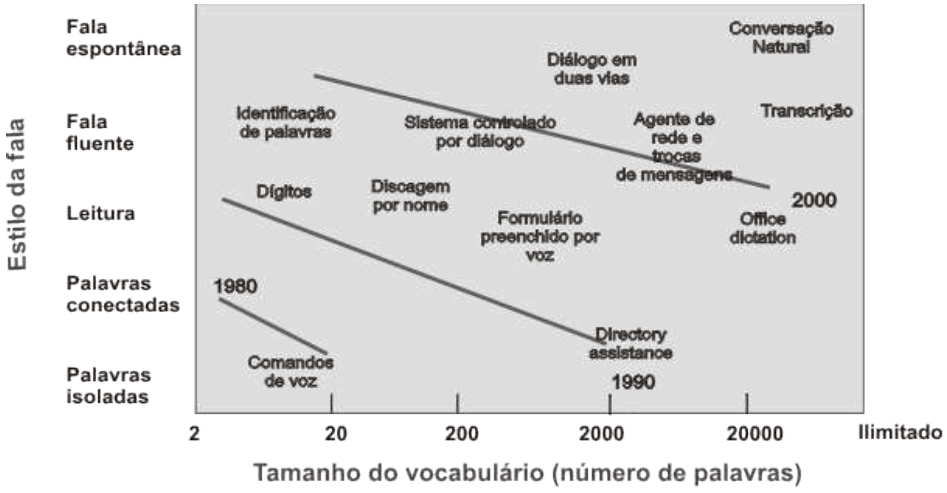


Figura 1.2: *Progresso da tecnologia na área de processamento de voz em termos de tamanho do vocabulário e forma da elocução (Adaptada de [2]).*

[20].

1.1 Justificativa e Relevância do Trabalho

A possibilidade de permitir a interação homem-máquina, a partir da voz, tem sido objeto de grande interesse, tanto do meio acadêmico quanto dos fabricantes de equipamentos eletrônicos [10, 11, 18]. Apesar de muitas pesquisas na área de PDSV terem permitido o desenvolvimento de sistemas de Resposta Vocal, Reconhecimento de Fala e Reconhecimento de Identidade Vocal bastante eficientes, requisitos de processamento ainda dificultam a implementação desses sistemas em dispositivos com pequeno poder computacional, como celulares, *palmtops* e eletrodomésticos [10, 11, 18].

Para permitir a implementação de sistemas de PDSV nesse contexto, alguns trabalhos sacrificam a eficiência no processo de reconhecimento em nome da redução da área e exigências computacionais.

Contudo, algumas técnicas de programação permitem a geração de códigos mais eficientes no que se refere à quantidade de operações a serem executadas pelo processador (p.ex.: substituição de uma multiplicação por deslocamento de bits), o que consequentemente também proporciona menor consumo de energia [21, 22]. Sendo assim, a partir da otimização de algoritmos, utilização de estruturas de *hardware* específicas e um paralelismo de instruções eficiente, é possível aumentar a velocidade e reduzir os recursos necessários, sem diminuir a taxa de reconhecimento.

O projeto de sistemas embarcados é bastante complexo, por envolver conceitos pouco analisados pela computação de propósito geral [10, 11]. Por exemplo, as questões de mobilidade, limite de consumo de energia, a baixa disponibilidade de memória, a necessidade de segurança e confiabilidade, a possibilidade de funcionamento em uma rede de comunicação e o curto tempo de projeto tornam o desenvolvimento de sistemas computacionais embar-

cados uma área de pesquisa bastante abrangente [23, 24]. Novos produtos têm uma vida cada vez mais curta, e atrasos de poucas semanas no lançamento de um produto podem comprometer seriamente os ganhos esperados. Com a automação do projeto de *hardware* caminhando na direção do reuso de plataformas e de componentes integráveis (*IP Cores*) em um SoC (*System on Chip*), a automação do projeto de *software* e sua integração com o projeto de *hardware* se torna o principal objetivo a ser alcançado para a diminuição do tempo total de projeto, sem sacrifício na qualidade da solução [24]. Também é essencial o reuso de componentes de *software*, de modo que o desenvolvimento do sistema concentre-se apenas na configuração e integração desses componentes [24, 25].

Tendo em vista o vigente interesse pelo desenvolvimento de aplicações em PDSV para sistemas embarcados e também as dificuldades enfrentadas nesse tipo de projeto, torna-se clara a importância da definição e implantação de ferramentas de *hardware* e *software* que otimizem essa produção. Muitos trabalhos podem ser encontrados para reconhecimento de palavras isoladas nesse contexto [10, 11, 15], ou ainda reconhecimento de fala contínua para idiomas diferentes do português brasileiro (PB) [26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36]. No entanto, poucos tratam de fala contínua para a língua portuguesa em sistemas embarcados.

Um impacto de grande relevância deste trabalho no âmbito científico consiste na adaptação das técnicas necessárias à realização de reconhecimento de fala contínua, projetados para a língua portuguesa, em sistemas com limitações de recursos de *hardware*, incluindo não só o desenvolvimento de *software*, como também, a modelagem para *hardware*.

No âmbito tecnológico e econômico, o desenvolvimento de uma nova forma que permita a comunicação entre o homem e dispositivos eletrônicos necessários ao seu cotidiano, gerará um novo tipo de interface homem-máquina, o que é de interesse não só da comunidade consumidora, como também das empresas fabricantes de tais dispositivos.

O impacto social se dará junto, por exemplo, à comunidade de pessoas portadoras de deficiência física visual e/ou motora, que poderão dispor de um sistema de reconhecimento de fala, que possibilitará uma melhoria na sua comunicação com as máquinas. Além disso, sistemas fixos e treinados para reconhecer a fala de um usuário, para responder a sua solicitação, bem como reconhecer quem está falando, poderão ser facilmente instalados em repartições públicas, hospitais, postos policiais e empresas prestadoras de serviços.

Além dos impactos e repercussões citados, o desenvolvimento deste trabalho tornará possível vislumbrar outras aplicações da comunicação vocal homem-máquina, visto que a máquina faz parte da vida do homem moderno e, portanto, a ciência deve buscar cada vez mais tornar mais fácil, atrativa e eficiente a interação do homem com as máquinas.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo principal deste trabalho consiste na modelagem da arquitetura de um sistema de reconhecimento de fala contínua para o português brasileiro, de forma a possibilitar sua implementação em um sistema embarcado com recursos computacionais limitados.

1.2.2 Objetivos Específicos

- Reutilizar a arquitetura desenvolvida no trabalho de mestrado [37], correspondente à etapa de pré-processamento do sinal de voz, com as funções de pré-ênfase e janelamento do sinal.
- Investigar o conjunto de características acústicas mais adequado à modelagem acústica, com base em parâmetros obtidos por análise LPC e técnicas derivadas dessa análise;
- Utilizar método estatístico para geração do modelo acústico, baseado em Modelos Ocultos de Markov com Densidades Contínuas;
- Investigar métodos para construção do modelo linguístico;
- Investigar a melhor configuração para a modelagem acústica e linguística com vistas ao compromisso entre a eficiência e custo computacional, ou seja, número de operações, elementos lógicos, ciclos de *clock*, entre outros;
- Modelar a arquitetura do sistema de reconhecimento de fala contínua para o português brasileiro com vistas à sua implementação em FPGA;

1.3 Metodologia

A metodologia empregada para desenvolvimento do trabalho apresentado nesta tese, consistiu nas etapas descritas a seguir.

1. Estudo comparativo entre técnicas para extração de características aplicadas ao reconhecimento de fala contínua;

   Nessa etapa, foi realizado o estudo do estado da arte das técnicas que estão sendo utilizadas para extração de características que mais se adequam ao reconhecimento de fala contínua. Nesse estudo, foram levantados os aspectos positivos e negativos de cada técnica no âmbito da aplicação pretendida (sistema embarcado).
   Foi analisada a eficiência de cada técnica no processo de reconhecimento *versus* a possibilidade de adaptação do algoritmo, de modo a permitir sua implementação em um

sistema embarcado.

2. Estudo comparativo entre técnicas para geração de modelos acústicos do sinal de voz;

   Assim como foi definido para o processo de extração de características, também foi realizada uma avaliação do estado da arte no que se refere às técnicas para geração de modelos acústicos de sinais de voz.
   Aqui também foi considerada a eficiência de cada técnica no processo de reconhecimento *versus* a possibilidade de adaptação do algoritmo.

3. Definição do modelo linguístico mais adequado para o português brasileiro;

   Foi realizado um estudo, com base no estado da arte, no que se refere às técnicas para geração de Modelos Linguísticos de sinais de voz.
   Mais uma vez, foi considerada a eficiência de cada técnica no processo de reconhecimento, além da possibilidade de adaptação do algoritmo que a implementa.

4. Avaliação das técnicas definidas nas etapas 1, 2 e 3, no que se refere à taxa de reconhecimento e ao custo computacional;

   Para um sistema de reconhecimento de fala eficiente, torna-se necessária a adpatação dos modelos acústicos e linguísticos para o português brasileiro. Nesta etapa, foram realizados testes em um *framework* para reconhecimento de fala (Sphinx), variando-se os principais parâmetros utilizados na modelagem acústica e linguística, analisando-se o impacto dos mesmos na taxa de reconhecimento e custo computacional.
   Uma vez que o objetivo geral deste trabalho é a implementação de um sistema de reconhecimento de fala contínua em um *hardware* de baixo custo, torna-se necessária uma avaliação da relação taxa de reconhecimento *versus* custo computacional necessário. Em alguns casos, ao simplificar o modelo utilizado, é possível obter-se uma economia dos recursos de *hardware* implicando em uma pequena diminuição da taxa de reconhecimento.

5. Adaptação dos algoritmos que implementam as técnicas escolhidas;

   Algumas técnicas de programação permitem a geração de um código mais eficiente no que se refere à quantidade de operações a serem executadas pelo processador (exemplo: substituição de uma multiplicação por deslocamento de bits).
   Nesta etapa, os algoritmos das técnicas selecionadas foram analisados e adaptados para atender os requisitos do contexto embarcado.

Ainda nesta etapa, foi realizada a validação em *software* das simplificações e adaptações realizadas.

6. Modelagem da arquitetura do sistema embarcado de reconhecimento de fala contínua para o português brasileiro;

   Com base nas adaptações obtidas na etapa anterior, foi realizada a modelagem completa do sistema pretendido.

7. Elaboração/Submissão de artigos para eventos nacionais e internacionais;

8. Elaboração da redação da Tese de Doutorado;

9. Defesa da Tese de Doutorado.

1.4 Organização do Documento

No Capítulo 2, são apresentados os conceitos relacionados ao processo de reconhecimento da fala, as etapas envolvidas, desde a caracterização do sinal de voz até as principais modelagens que auxiliam o processo de reconhecimento. No Capítulo 3, é apresentado o processo de modelagem e classificação de padrões da fala utilizando HMM. No Capítulo 4, é apresentado o processo de desenvolvimento de um sistema embarcado, as pecularidades desse tipo de projeto, assim como alguns trabalhos relacionados ao reconhecimento de fala nesse contexto. No Capítulo 5, é descrito o todo o processo de modelagem da arquitetura, incluindo a preparação da base de dados, a configuração do ambiente de testes, os experimentos realizados, a análise das etapas envolvidas no processo de reconhecimento, e os resultados que permitiram identificar a configuração da arquitetura modelada. No Capítulo 6, é apresentada a arquitetura do sistema embarcado de reconhecimento de fala contínua obtida ao final deste trabalho. Finalmente, no Capítulo 7 são apresentadas as considerações finais a respeito deste trabalho, citandos suas contribuições e sugestões para trabalhos futuros.

No Apêndice A, é apresentada a base de dados utilizada no trabalho. No Apêndice B são apresentados os valores dos 36 filtros utilizados neste trabalho. No Apêndice C são apresentadas as publicações relacionadas a este trabalho.

Capítulo 2

Caracterização e Modelagem do Sinal de Voz

Os principais tópicos envolvidos no reconhecimento de fala com vocabulário contínuo e grande são: modelagem linguística, representação léxica, modelagem e decodificação acústico-fonética [2, 9, 16, 18, 19].

A maioria dos trabalhos que vem sendo desenvolvida na área de reconhecimento automático da fala emprega o critério de maximização de P(W|X), sendo W = w_1, ..., w_k uma sequência de palavras e X = x_1, ..., x_t uma sequência de observações acústicas [8, 38]. No entanto, para se obter mais eficiência no processo de reconhecimento, é necessário modelar o processo de geração da fala, como apresentado na Figura 2.1 [8, 9, 38, 39].

Na Figura 2.1, M é a mensagem (conteúdo) que o locutor pretende transmitir. A mensagem M é concebida como uma sequência de palavras W, a partir de um canal linguístico, especificado por uma medida probabilística P(W|M). O canal linguístico é probabilístico. Dessa forma, existem diferentes maneiras de se expressar a mesma mensagem, algumas mais apropriadas que outras [2].

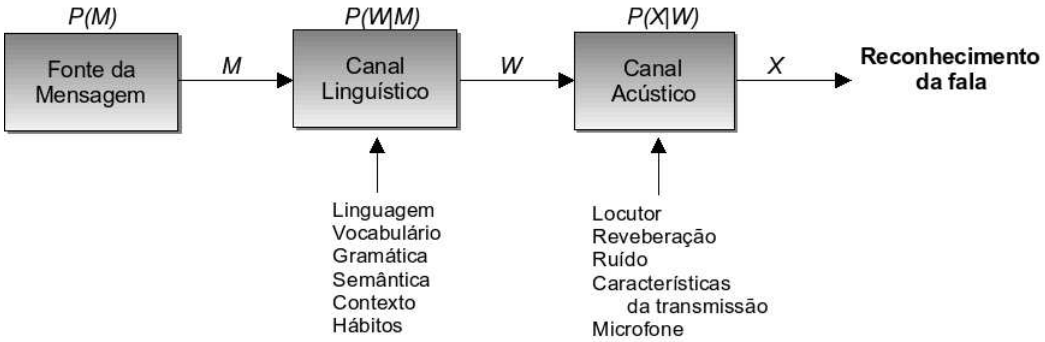


Figura 2.1: *Geração e reconhecimento da fala baseada na teoria da comunicação [2].*

A sequência de palavras W é então concebida, a partir do canal acústico, como uma sequência de sinais acústicos X. O canal acústico P(X|W) introduz variabilidade devido a diversas razões, incluindo aspectos do locutor e do ambiente acústico. Nenhum locutor consegue repetir exatamente a mesma forma de onda do sinal de voz, mesmo pronunciando

a mesma palavra, como também, nenhum locutor é igual a outro em termos de configuração de seu aparelho articulatório.

A sequência de sons irradiados pelos lábios do locutor se propaga em ondas acústicas no ambiente em que este se encontra. Essas ondas, combinadas aos sinais provenientes do ambiente, são captadas pelo microfone, quando, finalmente, são convertidas em um sinal elétrico. Esse sinal é propagado por meio de uma rota de transmissão (e.g. cabos, fios, rede telefônica), que é o canal acústico, produzindo o sinal X, o qual é recebido pelo sistema de reconhecimento. As características de todos esses processos variam substancialmente.

De acordo com esse modelo, o processo de reconhecimento de M consiste exatamente no caminho inverso ao processo de geração de M, e pode ser representado como a maximização da probabilidade a *posteriori* P(M|X) [2].

Os principais componentes de um sistema de reconhecimento de fala genérico são apresentados na Figura 2.2. Na figura, são apresentadas as fontes de alimentação necessárias ao sistema (amostras de fala e textuais, bem como o léxico de pronúncia [1]), como também as etapas de treinamento e decodificação. Os modelos acústico e linguístico, resultantes da etapa de treinamento, são utilizados como fonte de conhecimento na etapa de decodificação depois de as amostras de fala serem analisadas pelo *front-end* acústico. Nas próximas seções deste capítulo, será apresentado o papel de cada uma dessas modelagens no processo de reconhecimento.

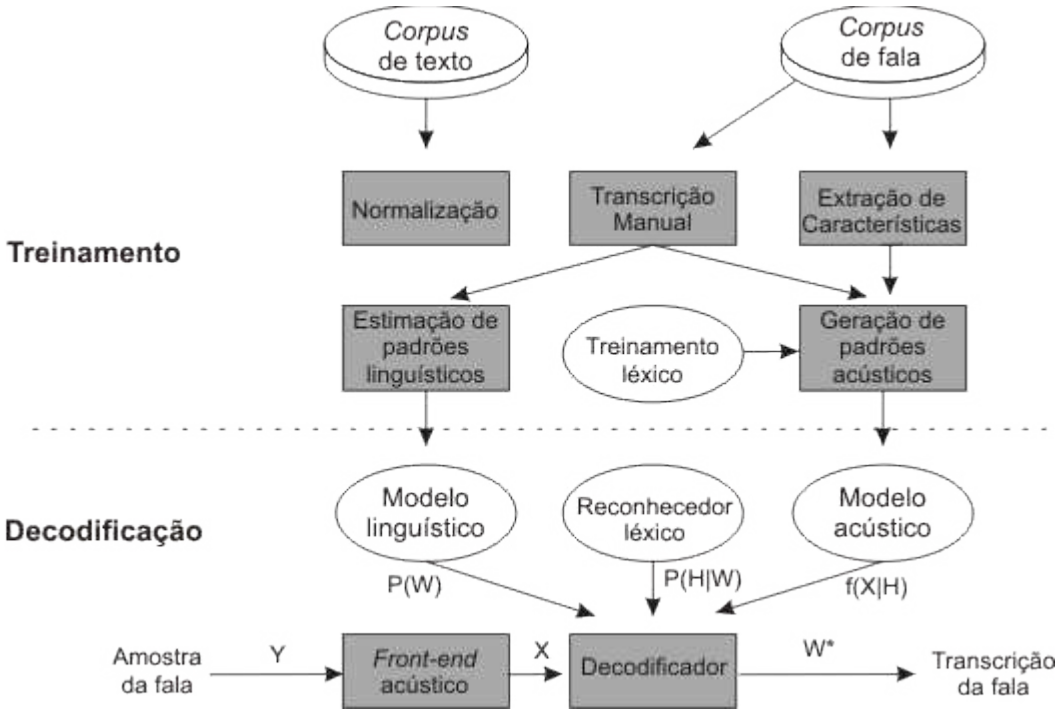


Figura 2.2: *Diagrama de um sistema genérico de reconhecimento de fala baseado em modelos estatísticos, incluindo as etapas de treinamento e decodificação [2].*

[1] Léxico de pronúncia, neste contexto, representa o conjunto de palavras definido no *corpus* (vocabulário) do sistema, juntamente com suas representações fonéticas. Vale salientar que, em alguns casos, uma mesma palavra pode ter mais de uma pronúncia, retratando assim, por exemplo, sotaques de diferentes regiões.

2.1 Modelagem Acústica

Um dos principais objetivos da modelagem acústica é lidar/tratar com a variabilidade presente em um sinal de voz. Essa variabilidade pode advir do contexto linguístico, ou pode estar associada ao contexto não linguístico, como por exemplo, o locutor (e.g. características físicas, forma da elocução, humor, etc.) ou ambiente acústico (e.g. ruído de fundo), além do canal de gravação (e.g. microfone, telefone, etc.).

A partir da extração de características acústicas do sinal, é possível obter elementos que possibilitam a geração dos padrões de fala. A definição de um bom conjunto de características é um processo complexo e essencial para que se obtenha bons resultados quanto ao reconhecimento dos padrões, podendo ser utilizados métodos de análise no domínio do tempo e no domínio da frequência. Análises no domínio do tempo transformam um sinal de voz em um conjunto de parâmetros, que variam bem mais lentamente que o sinal original. Isso permite um armazenamento, ou manipulação dos parâmetros relevantes, de uma maneira mais eficiente [40]. Contudo, parâmetros extraídos no domínio da frequência são mais consistentes do que aqueles extraídos no domínio do tempo [40], uma vez que, sentenças iguais repetidas pelo mesmo locutor podem obter uma grande variação no tempo enquanto que no domínio da frequência serão bem semelhantes. Além disso, o ouvido humano percebe muito mais sensivelmente variações espectrais, como distribuição da amplitude na frequência, do que fase e outros aspectos temporais [40].

Dois métodos de análise espectral predominam nos sistemas de reconhecimento de fala: o método de análise espectral LPC (*Linear Predictive Coding*) [1, 2, 41, 42, 43, 44] e o método de análise espectral por banco de filtros [45, 46, 47], os quais serão descritos ainda neste capítulo.

É importante ressaltar, que antes de extrair as características, o sinal de voz deve ser preparado para tal processo. Essa etapa de preparação do sinal é chamada de pré-processamento e será descrita na seção seguinte.

2.1.1 Pré-processamento

A etapa de pré-processamento é responsável pelo tratamento do sinal de voz com relação ao ambiente de gravação e canal de comunicação utilizado. O objetivo desse tratamento é reduzir efeitos indesejados incorporados ou presentes no sinal de voz, além de prepará-lo para as etapas seguintes do processo de reconhecimento. Dentre vários aspectos que podem ser tratados nessa etapa, podem ser citados [1, 14, 40]:

- Variações relacionadas ao estilo do falante;
- Ruído no ambiente, no meio de comunicação, etc;
- Variações no momento de gravação do sinal como, por exemplo, distância entre o locutor e o microfone;

- Considerações quanto ao tipo de unidade da fala a ser processada nas etapas posteriores.

O pré-processamento, realizado neste trabalho, inclui as etapas de pré-ênfase, segmentação e janelamento.

Pré-ênfase

A voz produzida pelo aparelho fonador humano sofre perdas durante sua passagem pelo trato vocal, inclusive na sua radiação através dos lábios. A distorção provocada pelos lábios produz uma queda na envoltória espectral de, aproximadamente, 6dB/oitava. Uma vez que o sinal de voz apresenta baixas amplitudes nas altas frequências, essa tendência as torna especialmente vulneráveis ao ruído, comprometendo o processo de reconhecimento. Para solucionar esse problema, é aplicado um filtro, de resposta aproximadamente +6dB/oitava, que ocasiona um nivelamento no espectro [1, 48]. A esse processo de tratamento do sinal de voz, dá-se o nome de pré-ênfase.

A função de transferência da pré-ênfase consiste em um sistema de primeira ordem fixo, cuja função é dada por:

$$H(z) = 1 - a.z^{-1} \quad , 0 \leq a \leq 1. \tag{2.1}$$

Nesse caso, a saída da pré-ênfase $s_p(n)$ está relacionada à entrada $s(n)$ pela equação diferença [1, 48]:

$$s_p(n) = s(n) - \alpha.s(n-1), \tag{2.2}$$

sendo:
$s_p(n)$ – amostra pré-enfatizada;
$s(n)$ – amostra original;
α – fator de pré-ênfase, $0,9 \leq \alpha \leq 1$.

Um valor típico usado é $\alpha = 0.95$, o que significa 20 dB de amplificação para as frequências mais altas. Na Figura 2.3, é ilustrado o processo de pré-ênfase para a palavra “aplausos”. É possível observar que, após a pré-ênfase, as amostras de frequências mais baixas sofrem uma atenuação, enquanto as de frequências mais altas, especialmente acima de 4 kHz, têm um ganho de amplitude.

Divisão em quadros e Janelamento

Um sinal é dito estacionário quando suas características estatísticas não variam com o tempo [1]. A segmentação consiste em particionar o sinal de voz em segmentos, selecionados por janelas ou quadros (*frames*) de duração perfeitamente definida. O tamanho desses segmentos

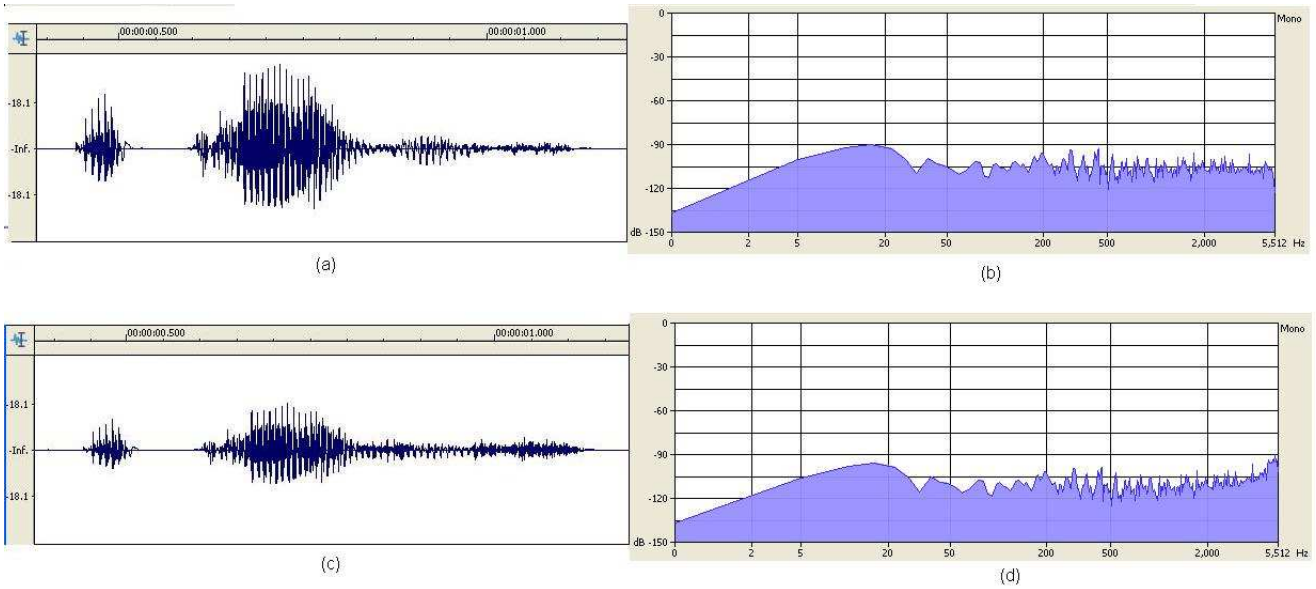


Figura 2.3: *Palavra "aplausos"– (a) Antes da Pré-ênfase e (b) seu espectro (c) após a Pré-Ênfase e (d) seu espectro.*

é escolhido dentro dos limites de estacionariedade do sinal (duração média de 16 a 32 ms) [1, 40].

Os tipos de janelas normalmente utilizados são [1, 13, 40, 48]:

- Janela retangular – o sinal é simplesmente particionado em blocos consecutivos de mesmo tamanho. Sua equação é dada por:

$$J(n) = \begin{cases} 1 & ,0 \leq n \leq N_A - 1 \\ 0 & ,\text{caso contrário.} \end{cases} \tag{2.3}$$

- Janela de *Hamming* – proporciona a manutenção das características espectrais do centro do quadro e a eliminação das transições abruptas das extremidades. Sua equação é dada por:

$$J(n) = \begin{cases} 0,54 - 0,46cos[2\pi n/(N_A - 1)] & ,0 \leq n \leq N_A - 1 \\ 0 & ,\text{caso contrário.} \end{cases} \tag{2.4}$$

- Janela de *Hanning* – assemelha-se à janela de Hamming, porém gera um reforço menor nas amostras do centro e uma suavização maior nas amostras da extremidade. Sua equação é dada por:

$$J(n) = \begin{cases} 2acos[\pi n/(N_A)] + b & ,0 \leq n \leq N_A - 1 \\ 0 & ,\text{caso contrário.} \end{cases} \tag{2.5}$$

sendo $2a + b = 1 \quad (0 \leq a \leq 0,25 \quad 0,5 \leq b \leq 1)$.

Na Figura 2.4, são apresentadas as janelas Retangular, de *Hanning* e de *Hamming*

e suas respectivas respostas em frequência. Como se pode observar, o lobo principal da janela retangular é aproximadamente a metade do lobo principal das janelas de *Hamming* e *Hanning*, enquanto os lobos secundários dessas últimas são bem menores que os da retangular. O primeiro lobo secundário da janela de *Hanning* é aproximadamente 20 dB maior que o da de *Hamming*, mas os lobos seguintes diminuem mais rapidamente que os da janela de *Hamming*.

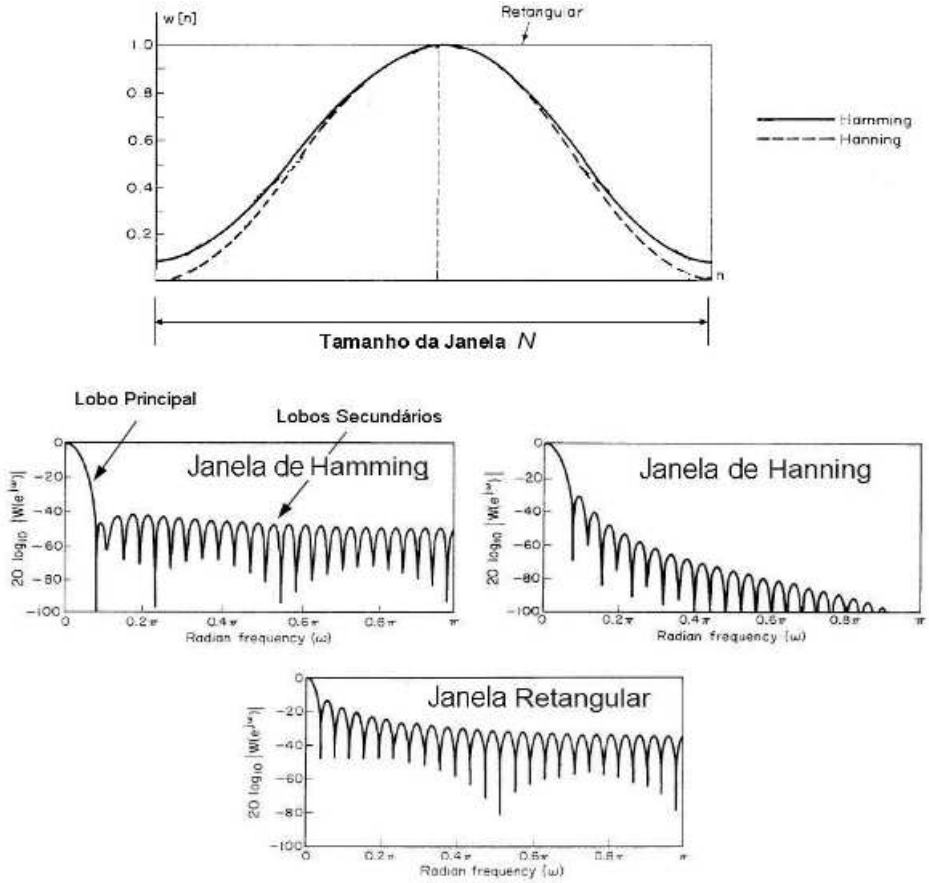


Figura 2.4: *Janelas de Hamming, Hanning e Retangular com suas respostas em frequência [3].*

A escolha da janela a ser utilizada é uma questão de avaliação da relação custo-benefício entre um lobo principal estreito e lobos secundários pequenos. Um lobo principal estreito melhora a resolução da frequência, o que permite que componentes estreitos muito próximos sejam separados. No caso da janela retangular, no entanto, esse lobo principal estreito vem seguido de lobos secundários altos, o que adiciona uma aparência ruidosa ao sinal, devido à interferência de harmônicos adjacentes, dificultando a discriminação de componentes de baixa amplitude. Já nas janelas de *Hamming* e *Hanning*, os lobos secundários menores permitem uma melhor detecção desses componentes. A janela de Hamming apresenta, porém, uma característica nem sempre desejável, que corresponde à atribuição de um peso muito baixo às amostras da extremidade. Entretanto, essas amostras podem representar eventos importantes de curta duração do sinal de voz e multiplicá-las por um peso baixo representa pouca atenção no processamento subsequente realizado no nível de blocos. Para assegurar

que a tais eventos seja dado o peso necessário, blocos adjacentes são sobrepostos de modo que um evento seja "coberto" por outros blocos. Muitos trabalhos utilizam uma sobreposição de 50% [46, 49]. Outra justificativa para a sobreposição das janelas é que esta proporciona uma variação mais gradual dos parâmetros entre janelas sucessivas [1, 48]. Para o contexto da produção da voz, as características apresentadas, referentes ao janelamento de Hamming, mostram que este tipo de janela é, portanto, mais eficiente, quando comparado às janelas Retangular e de *Hanning*, com uma boa aproximação da janela ideal. Assim sendo, essa foi a janela utilizada neste trabalho.

2.1.2 Análise por Predição Linear (*Linear Predictive Coding* - LPC)

A filosofia da predição linear está intimamente relacionada ao modelo de produção da voz (Figura 2.5), que mostra como o sinal de voz pode ser modelado como a saída de um sistema linear variante no tempo excitado por pulsos quase periódicos (para os sons sonoros), ou ruído aleatório (para sons não sonoros). As técnicas de predição fornecem um método robusto, realizável e correto para estimação dos parâmetros que caracterizam o sistema linear variante com o tempo [50]. A forma particular do modelo digital de produção de voz que é apropriada para a utilização da predição linear está descrita na Figura 2.5, sendo $s(n)$ o sinal de voz.

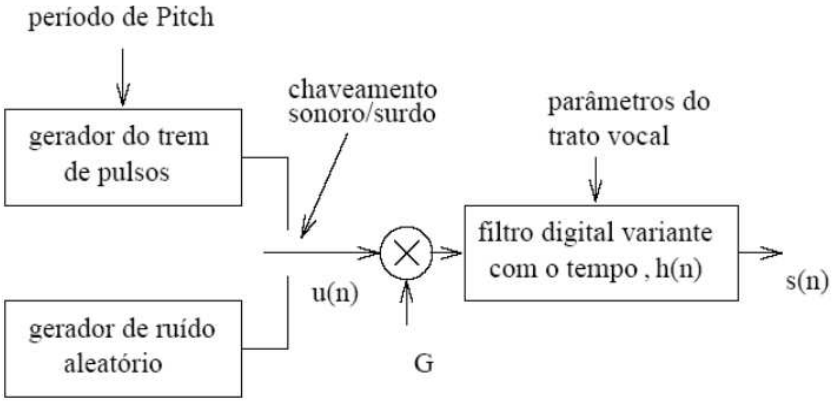


Figura 2.5: *Diagrama de blocos para o modelo simplificado de produção de voz.*

A idéia básica da predição linear reside no fato de que a voz amostrada pode ser aproximada como uma combinação linear das amostras de voz passadas e de valores presentes e passados de uma entrada hipotética de um sistema cuja saída é o sinal dado. No domínio da frequência, é equivalente a modelar o sinal por um espectro de pólos e zeros [50].

A partir da minimização da soma das diferenças quadradas (sobre um intervalo finito) entre as amostras reais da fala e as amostras obtidas com a combinação linear das primeiras, um conjunto único de coeficientes do preditor pode ser determinado. Os coeficientes do preditor são os coeficientes de ponderação usados na combinação linear.

Os métodos de predição linear estão disponíveis na literatura de engenharia há um longo tempo e têm sido vastamente empregados, principalmente em sistemas de controle, automa-

ção, telecomunicações e teoria da informação e codificação [1].

As técnicas de predição linear também podem ser aplicadas à quantização para reduzir a taxa de bits na representação digital do sinal de voz [1].

O princípio básico da predição linear leva a um conjunto de técnicas de análise que podem ser usadas para estimar parâmetros da fala. Esse conjunto geral de técnicas é frequentemente denominado de Análise por Codificação Preditiva Linear ou Análise LPC (*Linear Predictive Coding*) [50].

Muitos sistemas de reconhecimento de fala e de locutor têm, tradicionalmente, utilizado os parâmetros obtidos da análise LPC, em virtude das vantagens que esses propiciam em termos de generalização da envoltória espectral, independência do *pitch* das harmônicas, e a sua habilidade para modelar, razoavelmente bem, os picos espectrais [51].

O principal problema associado à análise por predição linear é determinar um conjunto de coeficientes do preditor diretamente a partir do sinal de voz, a fim de se obter uma boa estimativa das propriedades espectrais do sinal de voz. Devido à natureza variante no tempo do sinal de voz, os coeficientes do preditor devem ser estimados em segmentos de curtos intervalos de tempo, nos quais as características estatísticas do sinal podem ser consideradas estacionárias. Esses coeficientes podem ser obtidos a partir da análise LPC, denominados coeficientes LPC, ou a partir de técnicas derivadas dessa análise. Dentre os coeficientes utilizados, destacam-se: coeficientes LPC, Cepestrais, Cepestrais Ponderados, Delta Cepestrais e Delta Cepestrais Ponderados. Esses parâmetros visam capturar informação espectral suficiente para permitir a tarefa de reconhecimento [14, 52].

A seguir, serão descritas as formas de obtenção dos coeficientes citados.

Coeficientes LPC

Vários são os métodos conhecidos para a determinação dos coeficientes LPC. Dentre esses: o método da covariância [53]; o método da autocorrelação [54]; a formulação do filtro inverso [1]; a formulação da estimação espectral [1]; a formulação da máxima verossimilhança [1] e a formulação do produto interno [1].

Um estudo dos vários métodos e a comparação entre eles podem ser encontrados em [1]. No modelo descrito na Figura 2.5, os efeitos da radiação, trato vocal, e excitação glotal são representados por um filtro digital variante no tempo cuja função de transferência, $H(z)$, tem a seguinte forma [1]:

$$H(z) = \frac{S(z)}{U(z)} = \frac{G}{1 - \sum_{k=1}^{K} c_k z^{-k}}, S(z) = U(z)H(z), \tag{2.6}$$

sendo:

$S(z)$ – Transformada-z da sequência de voz $s(n)$;
$U(z)$ – Transformada-z do sinal de excitação $u(n)$;
c_k – coeficientes LPC;
K – ordem da predição (número de coeficientes).

O sistema é excitado por um trem de pulsos para sons sonoros ou por uma sequência de ruído aleatório para sons não sonoros. Assim, os parâmetros do modelo são: classificação sonoro/não sonoro, parâmetro de ganho (G) e os coeficientes do filtro digital (c_k). Esses parâmetros variam muito pouco em curtos intervalos de tempo [1].

A maior vantagem do modelo é que o ganho, (G) e os coeficientes do filtro, (c_k), podem ser estimados, de forma computacionalmente eficiente, pelo método de predição linear.

Para o sistema da Figura 2.5, as amostras de voz, $s(n)$, são relacionadas com a excitação, $u(n)$, pela equação diferença [1]:

$$s(n) = \sum_{k=1}^{K} c_k s(n-k) + Gu(n). \tag{2.7}$$

Uma predição linear com coeficientes de predição, c_k, é definida como um sistema cuja saída é:

$$\tilde{s}(n) = \sum_{k=1}^{K} c_k s(n-k). \tag{2.8}$$

O erro de predição, $e(n)$, é definido como:

$$e(n) = s(n) - \tilde{s}(n) = s(n) - \sum_{k=1}^{K} c_k s(n-k). \tag{2.9}$$

Para formular o problema, inicialmente, é selecionado um segmento do sinal de voz por intermédio de uma janela de comprimento finito e igual a N_A. A melhor escolha do valor de N_A permite uma boa aproximação às hipóteses de ergodicidade e estacionariedade no sentido amplo [1]. Em virtude da inércia dos articuladores, é intuitivo que o sinal de voz possa ser considerado estacionário em intervalos apropriados, de curta duração.

Nas próximas equações, $vs(n)$ corresponderá ao segmento selecionado e ponderado pela janela sendo, assim, nulo no intervalo $n < 0$ e $n > N_A$. A origem do eixo "n" será estabelecida no início de cada segmento, para simplificar as notações.

A Equação 2.7, modificada pela janela, será escrita no domínio do tempo como [55]:

$$vs(n) = Gu(n) + \sum_{k=1}^{K} c_k vs(n-k). \tag{2.10}$$

Como dito anteriormente, a idéia principal da Predição Linear consiste em aproximar cada amostra do sinal de voz pela combinação linear de amostras passadas do sinal. Sendo K o número de amostras passadas utilizadas na combinação linear, pode-se formalizar a aproximação da amostra genérica $vs(n)$ pela relação [55]:

$$\tilde{vs}(n) = \sum_{k=1}^{K} c_k vs(n-k). \tag{2.11}$$

Dado que $\tilde{vs}(n)$ é a aproximação de $vs(n)$ e c_k é o $k-$ésimo coeficiente da combinação linear; $\tilde{vs}(n)$ é normalmente denominada a estimativa ou predição de ordem K da amostra $vs(n)$.

O erro de predição da cada amostra, $e(n)$, é definido por:

$$e(n) = vs(n) - \tilde{vs}(n) = vs(n) - \sum_{k=1}^{K} c_k vs(n-k). \tag{2.12}$$

E o erro quadrático, $Erro(n)$, acumulado em todo o segmento é dado por:

$$Erro(n) = \sum_{n=-\infty}^{\infty} e(n)^2. \tag{2.13}$$

Como o segmento de voz é nulo para $n < 0$ e para $n > N_A$, o erro de predição (Equação 2.12) é, portanto, nulo para $n < 0$ e $n > N_A + K - 1$. A partir dessa consideração, e substituindo a Equação 2.12 na Equação 2.13, obtém-se:

$$Erro(n) = \sum_{n=0}^{N_A+K-1} [vs(n) - \sum_{k=1}^{K} c_k vs(n-k)]^2. \tag{2.14}$$

O conjunto de coeficientes c_k que minimiza $Erro(n)$ é obtido a partir de:

$$\frac{\partial[Erro(n)]}{\partial[c_k]} = 0, \quad 1 \leq k \leq K. \tag{2.15}$$

Com a substituição da Equação 2.14 em Equação 2.15 e a realização das K derivadas

parciais, chega-se ao seguinte sistema de equações lineares:

$$\sum_{k=1}^{K} c_k R_r(|i-k) = R_r(i), \quad i \leq i \leq K. \tag{2.16}$$

com:

$$R_r(k) = \sum_{n=0}^{N_A-K-1} vs(n)vs(n+k), \tag{2.17}$$

denominada função de autocorrelação a curto prazo. As Equações 2.16 e 2.17, conhecidas como Equação de Wiener-Hopf, podem ser vistas mais facilmente se colocadas da seguinte forma (forma matricial) [5, 55]:

$$\begin{vmatrix} R_r(0) & R_r(1) & \cdots & R_r(K-1) \\ R_r(1) & R_r(0) & \cdots & R_r(K-2) \\ R_r(2) & R_r(1) & \cdots & R_r(K-3) \\ \cdots & \cdots & \cdots & \cdots \\ R_r(K-1) & R_r(K-2) & \cdots & R_r(0) \end{vmatrix} \begin{vmatrix} c_1 \\ c_2 \\ c_3 \\ \cdots \\ c_k \end{vmatrix} = \begin{vmatrix} R_r(1) \\ R_r(2) \\ R_r(3) \\ \cdots \\ R_r(K) \end{vmatrix} \tag{2.18}$$

Os coeficientes c_k do preditor são determinados a partir da solução das Equações 2.16 e 2.17 (ou da Equação 2.18) e são os coeficientes c_k do filtro $H(z)$ da Figura 2.5.

Utilizando a simetria da matriz de autocorrelação, pode-se utilizar algoritmos recursivos bastante eficientes para solução do sistema, a exemplo do algoritmo de Levinson-Durbin [55].

Após a estimação dos coeficientes do polinômio, falta determinar o ganho, G, expresso da seguinte forma [1]:

$$G = [R_r(0) - \sum_{k=1}^{K} c_k R_r(k)]^{\frac{1}{2}}, \tag{2.19}$$

sendo $R_r(k)$ a função de autocorrelação calculada com atraso k. Esta relação é válida tanto para excitação periódica (sons sonoros) quanto para excitação turbulenta (sons surdos) do modelo.

Coeficientes Cepestrais

Os coeficientes Cepestrais são usados para descrever a envoltória espectral do sinal de voz a curtos intervalos de tempo. O *cepstrum* é a transformada inversa de Fourier do logaritmo do espectro do sinal a curtos intervalos de tempo. A partir da operação logaritmo, a função de transferência do trato vocal e da fonte de voz são separadas [5, 56].

Uma das principais vantagens dos coeficientes Cepestrais reside no fato de que estes são geralmente descorrelacionados e isso gera covariâncias diagonais a serem utilizadas nos HMM [5].

Existem duas formas de obtenção dos coeficientes Cepestrais [5]: coeficientes Cepestrais FFT e coeficientes Cepestrais LPC.

Na análise cepestral FFT é aplicada diretamente ao sinal de voz uma transformada inversa rápida de Fourier. O $i-$ésimo cepstrum, $ce_i(n)$, é calculado por [5]:

$$ce_i(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} log_{10}|X_i(e^{jw})|e^{jwn}dw, \tag{2.20}$$

em que $-\infty < n < \infty$ e X_i representa o i-ésimo bloco do espectro de potência do sinal de voz a curtos intervalos de tempo.

Na análise cepestral LPC, a transformada z é aplicada no sinal de voz modelado pela análise LPC. Os coeficientes Cepestrais, do espectro obtido da análise LPC, podem ser calculados recursivamente, a partir dos coeficientes LPC, c_i, por [5, 52]:

$$\begin{aligned} ce_i(1) &= c_i(1), \\ ce_i(n) &= c_i(n) + \sum_{j=1}^{n-1}(1 - \frac{j}{n})c_i(j)ce_i(n-j), \quad 1 < n \leq K, \end{aligned} \tag{2.21}$$

em que n é o índice do coeficiente e i o índice do bloco de amostras.

O uso desta relação recursiva leva a uma computação eficiente dos coeficientes Cepestrais, $ce_i(n)$, e evita a fatoração polinomial. Uma vez que $ce_i(n)$ tem duração infinita, o vetor de características, de dimensão K, é constituído das componentes $ce_i(1)$ a $ce_i(K)$, as quais são as mais significativas devido ao "decaimento" da sequência com o aumento de n. Mesmo com esse truncamento, o erro médio quadrático entre dois vetores de coeficientes Cepestrais é aproximadamente igual ao erro médio quadrático entre os logaritmos do espectro dos correspondentes filtros LPC. Dessa forma, tem-se uma boa medida da diferença na envoltória espectral dos blocos de amostras a partir dos quais os coeficientes Cepestrais foram obtidos [52].

Análise a partir de banco de filtros e coeficientes mel-cepstrais

A análise mel-cepstral vem progressivamente substituindo a forma tradicional de utilização de parâmetros cepstrais previamente descritos [45]. Os coeficientes mel-cepstrais surgiram devido a estudos na área de psicoacústica que mostraram que a percepção humana das frequências de tons puros ou de sinais de voz não segue uma escala linear. Diante disso, foram definidas frequências subjetivas de tons puros, da seguinte forma: para cada tom com frequência f, medida em Hz, define-se um tom subjetivo medido em uma escala chamada

mel. O mel, então, é uma unidade de medida da frequência percebida de um tom.

Como referência, definiu-se a frequência de 1 kHz, com potência 40 dB acima do limiar mínimo de audição do ouvido humano, como 1000 mels [48]. Os outros valores subjetivos foram obtidos por meio de experimentos, em que se pedia a ouvintes que ajustassem a frequência física de um tom, até que a frequência percebida fosse igual a duas vezes a frequência de referência; depois, 10 vezes a frequência de referência e assim por diante. Essas frequências teriam os valores de 2000 mels, 10000 mels e, assim, sucessivamente. O mesmo processo era efetuado na outra direção, ou seja, metade do tom de referência, um décimo do tom de referência, etc. Essas frequências teriam valores de 500 mels, 100 mels, etc.

Na Figura 2.6 é apresentado um gráfico da frequência fundamental subjetiva de tons em função da frequência [57]. A curva superior mostra a relação entre as mesmas em uma escala linear. Pode-se observar que a frequência fundamental subjetiva, em mels, cresce menos rapidamente à medida que há um aumento linear na frequência. A curva inferior, por outro lado, mostra a frequência fundamental subjetiva em função da frequência em uma escala logarítmica. Também é possível observar, que a frequência fundamental subjetiva é essencialmente linear para frequências inferiores a 1000 Hz.

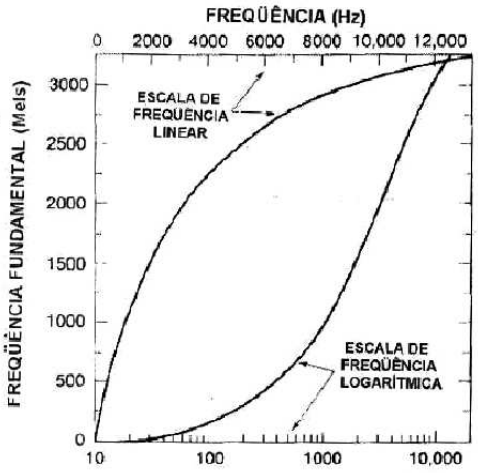


Figura 2.6: *Percepção subjetiva da frequência fundamental de sons sonoros.*

É possível traçar uma comparação entre a frequência real (medida em Hz) e a frequência percebida (medida em *mels*). O mapeamento entre a escala de frequência real, em Hz, e a escala de frequências percebida, em *mel*, é aproximadamente linear abaixo de $1000Hz$ e, logarítmica, acima. Pode-se definir uma função para mapeamento da frequência acústica f (em Hz) para uma escala de frequências percebidas Mel (em *mels*) como definido na Equação 2.22 e cujo gráfico é apresentado na Figura 2.7 [40].

$$F_{mel} = 2595 \cdot log_{10}(1 + \frac{F_{linear}(Hz)}{700}), \tag{2.22}$$

sendo F_{linear} a frequência linear (em Hz) e F_{mel} a frequência percebida (em mel).

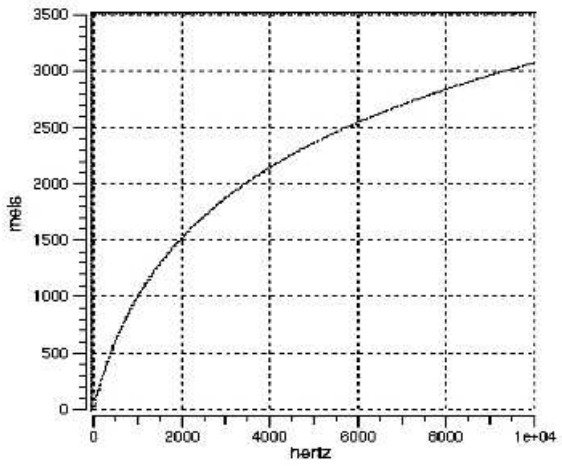


Figura 2.7: *Mapeamento de frequências entre escala mel e linear.*

Um outro critério importante relacionado ao conteúdo de frequência de um sinal é a *banda crítica* [58]. Alguns experimentos mostraram que algumas frequências, em determinados sons, não podem ser individualmente identificadas pelo sistema auditivo humano, para certas faixas de frequências. Quando uma componente de frequência não pertence a essa faixa de frequências, denominada de banda crítica, ela pode ser identificada. Uma explicação bem aceita é que a percepção, pelo sistema auditivo, de uma determinada frequência é influenciada pela energia da banda crítica em torno da frequência em questão. O valor dessa banda varia nominalmente de 10 a 20% da frequência central do som, começando em torno de 100 Hz para frequências abaixo de 1 kHz e aumentando em escala logarítmica, acima desse valor (Figura 2.8).

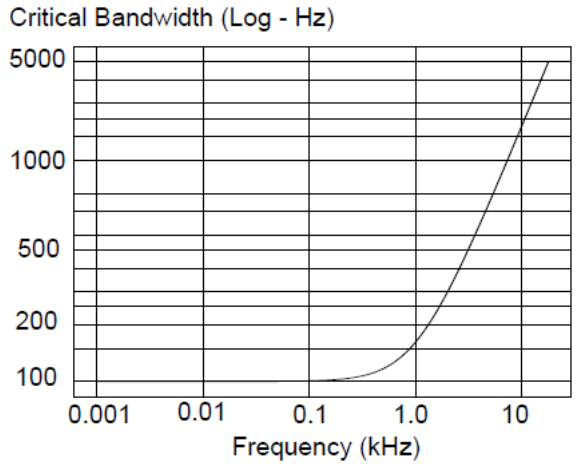


Figura 2.8: *Banda Crítica.*

Dessa forma, algumas modificações foram realizadas na representação espectral de um sinal, de forma a favorecer sistemas de reconhecimento de fala e de locutor. Tais modificações consistiram na ponderação da escala de frequência para a escala mel e na incorporação do conceito de *banda crítica*. Ou seja, é utilizado o logaritmo da energia total das bandas críticas em torno das frequênicas *mel*. A aproximação mais utilizada para esse cálculo é a utilização

de um banco de filtros triangulares (2.9), espaçados uniformemente em uma escala não linear (escala Mel).

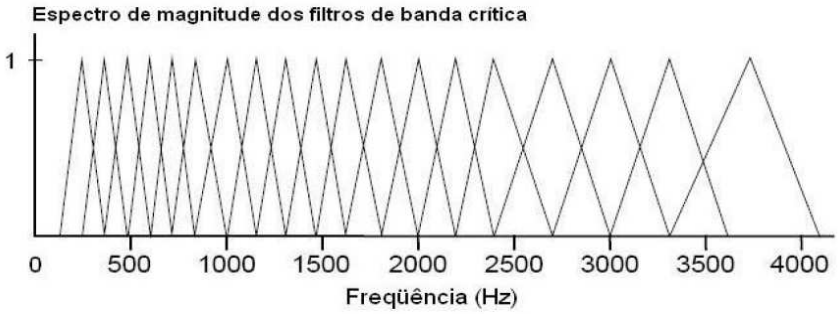


Figura 2.9: *Banco de filtros digitais na escala mel.*

Na Figura 2.10, é ilustrado o processo de obtenção dos coeficientes mel-cepstrais [46].

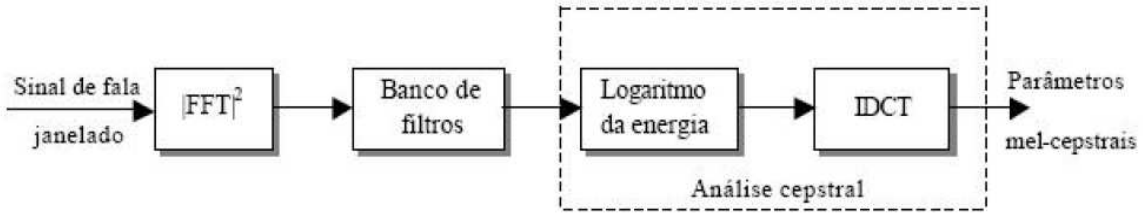


Figura 2.10: *Processo de obtenção dos coeficientes mel-cepstrais.*

A extração dos coeficientes MFCC ocorre após a etapa de pré-processamento, quando é realizada a divisão do sinal de voz $s(n)$ em janelas. Para cada janela m, estima-se o espectro $S(w,m)$, utilizando a FFT, cujo espectro de magnitude é dado por:

$$|S(w,m)| = (Re[S(w,m)]^2 + Im[S(w,m)]^2)^{\frac{1}{2}}. \tag{2.23}$$

O espectro modificado $P(i), \quad i = 1, 2, \ldots, N_f$, consistirá na energia de saída de cada filtro expresso por:

$$P(i) = \sum_{k=0}^{N/2} |S(k,m)|^2 H_i\left(k\frac{2\pi}{N}\right). \tag{2.24}$$

sendo N o número de pontos da FFT, N_f o número de filtros triangulares, $|S(k,m)|$ o módulo da amplitude na frequência do k-ésimo ponto da m-ésima janela e $H_i(w)$ é a função de transferência do i-ésimo filtro triangular, definido por:

$$H_i(w) = \begin{cases} \frac{1}{k_i - k_{i-1}}(w - k_{i-1}) & k_{i-1} \leq w \leq k_i \\ \frac{1}{k_i - k_{i+1}}(w - k_{i+1}) & k_i \leq w \leq k_{i+1} \end{cases} \tag{2.25}$$

sendo, k_i o i-ésimo centro, $k_0 = 0$, e w é uma escala ajustada de acordo com o número de pontos da FFT e expressa por:

$$w = k\frac{2\pi}{N} \quad 0 \leq k \leq N/2. \tag{2.26}$$

Em seguida, define-se o conjunto de pontos $E(k)$ por:

$$E(k) = \begin{cases} log[P(i)] & , k = k_i \\ 0 & ,\text{qualquer outro } K \in [0, N-1] \end{cases} \tag{2.27}$$

Por fim, os coeficientes mel-cepstrais $c_mel(n)$ são obtidos aplicando-se a Transformada Inversa de Fourier (IFFT), com base na seguinte equação:

$$c_{mel}(n) = \frac{1}{N}\sum_{k=0}^{N-1} E(k)e^{j\left(\frac{2\pi}{N}\right)kn} \quad n = 1, 2, \ldots, N_c \tag{2.28}$$

sendo N_c o número de coeficientes desejado.

Contudo, essa etapa pode ser simplicada aplicando-se a IDCT (*Inverse Discrete Cosine Transform*), já que $E(k)$ é simétrico em relação a $N/2$ (ou $\pi/2$) e lembrando que:

$$e^{j\left(\frac{2\pi}{N}\right)kn} = cos\left(\frac{2\pi}{N}kn\right) + jsen\left(\frac{2\pi}{N}kn\right), \tag{2.29}$$

resulta que os termos em seno da 2.28 se cancelam, gerando a equação:

$$c_{mel}(n) = \frac{1}{N}\sum_{k=0}^{N-1} E(k)cos\left(\frac{2\pi}{N}kn\right). \tag{2.30}$$

Ainda usando a simetria e observando que

$$E(0) = E(N/2), \tag{2.31}$$

obtém-se a expressão

$$c_{mel}(n) = \frac{2}{N}\sum_{k=1}^{\frac{N}{2}-1} E(k)cos\left(\frac{2\pi}{N}kn\right). \tag{2.32}$$

Sabendo-se que no intervalo $0 \leq k \leq (N-2)/2$ existirão apenas N_f termos diferentes de zero, que sãos os correspondentes aos centros dos filtros, e eliminando-se o fator de escala $2/N$, a equação 2.32 pode ser simplificada, chegando-se à expressão final para os coeficientes MFCC, dada por:

$$c_{mel}(n) = \sum_{i=1}^{N_f} E(k_i) cos\left(\frac{2\pi}{N} k_i n\right) \quad n = 1, 2, \ldots, N_c \tag{2.33}$$

sendo N_c o número de coeficientes mel-cepstrais desejado, N_f o número de filtros e k_i o centro do i-ésimo filtro.

Além dos mel-cepstrais, é comum os sistemas utilizarem as derivadas temporais como componentes adicionais no vetor de atributos, colocando-as em um patamar de igual relevância ao dos coeficientes em si [59, 60]. O objetivo é refletir melhor as mudanças dinâmicas dos MFCC no tempo. As componentes de primeira derivada, conhecidas como $\Delta-$Cepstrum ou $\Delta-$MFCC, representam a velocidade com que o espectro mel-cepstral varia e são facilmente computadas calculando a diferença entre coeficientes de m índices no passado e no futuro do tempo levado em consideração. É de grande valia também as componentes de aceleração do espectro mel-cepstral, por isso são utilizadas as derivadas de $\delta-$MFCC, chamadas de $\Delta\Delta-$Cepstrum ou $\Delta\Delta-$MFCC.

A primeira e segunda derivadas de MFCC são obtidas a partir das Equações 6.14 e 6.15, respectivamente:

$$\Delta c_{mel}[n] = c_{mel}[n+m] - c_{mel}[n-m], \tag{2.34}$$

$$\Delta\Delta c_{mel}[n] = \Delta c_{mel}[n+m] - \Delta c_{mel}[n-m], \tag{2.35}$$

sendo n o índice o coeficiente em questão e m, comumente no intervalo $1 \leq m \leq 4$ [59].

2.2 Modelagem Linguística

Modelos linguísticos (ML) capturam regularidades na linguagem falada e são utilizados no reconhecimento para estimar a probabilidade da sequência de palavras. Enquanto restrições gramaticais feitas manualmente podem ser utilizadas para pequeno ou médio vocabulário, para vocabulário grande e contínuo são utilizadas abordagens *data-driven* [2] [2]. Um modelo Linguístico Estatístico é a distribuição probabilidade $P(W)$ de todas sentenças possíveis , de acordo com a regra de Bayes na Equação 2.36 [61].

[2] *Data-driven* - interpretações vão sendo realizadas à medida que os dados são analisados.

$$\begin{aligned}\widehat{W} &= \underset{W}{argmax} P(W|O) \\ &= \underset{W}{argmax} \frac{P(O|W)P(W)}{P(O)} \\ &= \underset{W}{argmax} P(O|W)P(W), \end{aligned} \tag{2.36}$$

sendo $P(W)$ a probabilidade a priori que o locutor pronuncie W, que é independente dos dados acústicos e é determinada pelo modelo linguístico, e $P(O|W)$ é a probabilidade que o locutor produza o dado acústico O se W é a sequência de palavras que o locutor deseja emitir.

A maioria dos modelos linguísticos atuais decompõe a probabilidade de uma sentença W no produto das probabilidades condicionais de acordo com:

$$P(W) = P(w_1, w_2, ..., w_n) = \prod_{i=1}^{n} P(w_i|w_1, w_2, ..., w_{i-1}) = \prod_{i=1}^{n} P(w_i|h_i), \tag{2.37}$$

sendo w_i a i-ésima palavra na sentença, e $h_i = w_1, w_2, ..., w_{i-1}$ o histórico da palavra. Sistemas para reconhecimento de vocabulário grande e contínuo podem utilizar um ou mais modelos linguísticos. Embora esses modelos sejam estáticos, o processo de escolha do modelo a ser utilizado pode ser dinâmico [2].

2.2.1 Modelagem Linguística: N-gram

O método linguístico mais popular é o chamado modelo n-gram, que objetiva capturar as restrições sintáticas e semânticas da língua a partir da estimativa da frequência de sequência de n palavras [9, 38, 39, 62, 63].

Para se obter um modelo n-gram eficiente, torna-se necessária a utilização de um *corpus* apropriado de texto processado. Como pode ser observado na Figura 2.2, modelos linguísticos, usualmente, são estimados por meio de transcrições manuais do *corpus* de texto. Para garantir modelos corretos, o *corpus* deve ser o mais representativo possível do áudio de entrada esperado.

Para uma sequência de palavras $W = w_1, w_2, ..., w_n$ de n palavras, pode-se escrever a definição do modelo linguístico *N-gram* como:

$$P(W) = P(w_1, w_2, ..., w_n) = \prod_{i=1}^{n} P(w_i|w_1, w_2, ..., w_{i-1}). \tag{2.38}$$

No modelo *N-gram*, o valor de N é escolhido com base no compromisso entre a confiabilidade do treinamento e a eficiência do modelo. A probabilidade da próxima palavra w_i de-

pende da história $h_i = (w_i|w_0, ..., w_{i-1})$ das palavras que foram pronunciadas anteriormente. Com essa fatoração, a complexidade do modelo de linguagem cresce exponencialmente com o comprimento da história. De modo a obter um modelo mais prático e simplificado, a história de palavras já pronunciadas é truncada, de modo que apenas alguns termos são utilizados para calcular a probabilidade de uma palavra seguir a palavra atual.

O modelo *3-gram* mais comum é obtido quando $N = 3$ na Equação 2.38, que é usado com grande corpo de treinamento (milhões de palavras ou mais). O modelo *2-gram* é frequentemente usado para pequeno conjunto de dados de treinamento. O modelo linguístico monograma, *1-gram*, correspondente a $P(w_i|h_i) \approx P(w_i)$, e o modelo *flat* (ou *0-gram*, quando $P(w_i|h_i) \approx constante$) apresenta grande limitação na estimação das restrições sintáticas e semânticas da sequência de palavras da linguagem.

2.2.2 Árvore de Decisão

Árvores de decisão é um dos métodos mais populares de agrupamento e tem sido utilizado em diversas áreas [4, 64]. É utilizada uma abordagem *top-down* para separar as amostras, que estão inicialmente em um único grupo, em grupos menores, por meio de perguntas sobre tais amostras. Na Figura 2.11, é ilustrada a idéia das árvores de decisão.

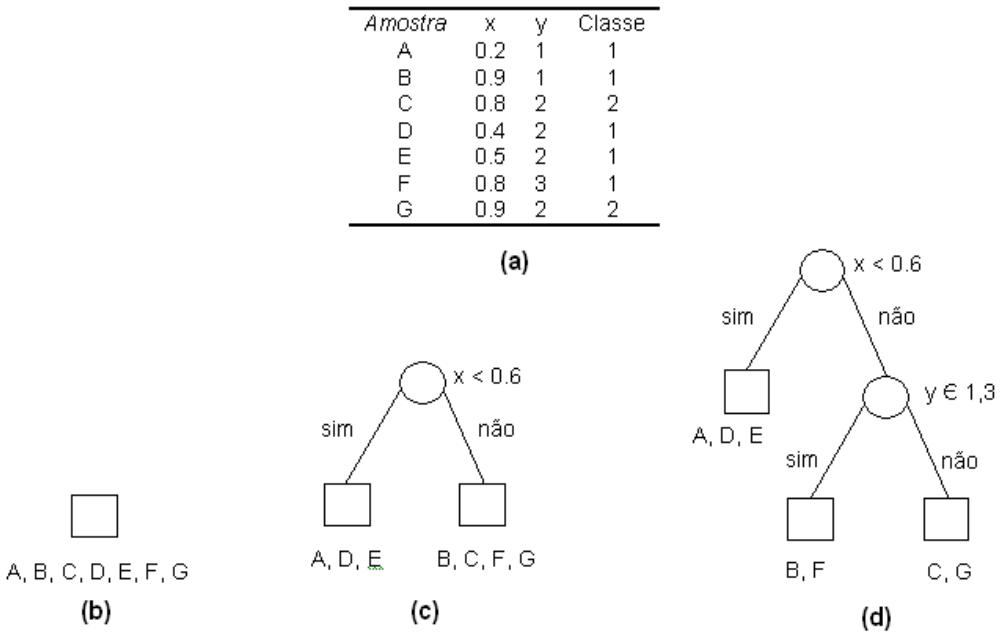


Figura 2.11: *Exemplo de uma árvore de decisão[4].*

Inicialmente, todas as amostras (A, B, C, D, E, F, G possuem as propriedades x de y) são colocadas no nó raiz (Figura 2.11(b)). A partir daí, as perguntas sobre as propriedades das amostras vão sendo realizadas, e de acordo com as respostas, novos nós filhos vão sendo criados e as amostras sendo colocadas nesses (Figura 2.11(c)). Dando continuidade ao processo, novas perguntas vão sendo realizadas nos nós filhos e, novamente as amostras vão sendo separadas (Figura 2.11(d)). O resultado é uma árvore, cujos nós que não são folhas

contêm perguntas sobre as amostras, e os nós folhas contêm amostras que tiveram as mesmas respostas, durante o caminho percorrido da raiz até a folha. É importante salientar, que a escolha das perguntas deve ser bem elaborada, de modo que se obtenham folhas puras, ou seja, que contenham amostras que realmente pertençam à mesma classe.

Na Figura 2.11, as perguntas possuíam apenas duas respostas (sim ou não), o que produz uma árvore de decisão binária, contudo, outros tipos de árvores podem ser utilizados.

Algoritmos de árvore de decisão foram desenvolvidos e aplicados em modelagem linguística pela primeira vez em [65]. Uma árvore de decisão separa a história h (palavras pronunciadas anteriormente) de uma palavra w com base em questões binárias a cada nó interno da árvore. A cada folha, os dados de treinamento são então utilizados para construir a distribuição de probabilidade $P(w|h)$ sobre a próxima palavra [64]. Para melhorar o grau de confiabilidade da estimação, a distribuição da folha atual é interpolada com as distribuições dos nós internos da árvore de decisão, encontrados ao longo do caminho até o nó.

Árvores de decisão têm sido muito utilizadas na modelagem linguística para relacionar/ligar os parâmetros do HMM, bem como, melhorar o desempenho dos modelos *n-gram* [64, 66]. O maior desafio técnico utilizando árvores de decisão é o imenso espaço de históricos de palavras e possíveis perguntas. Para superar essas dificuldades, as classes de perguntas são fortemente restritas evitando-se redundância nas questões, de forma a se obter maior eficiência no processo de busca e decisão. Contudo, árvores de decisão também podem ser utilizadas na modelagem acústica.

2.2.3 Gramática livre de contexto como Modelagem Linguística

Uma gramática livre de contexto (CFG – *context-free grammar*) é definida por um vocabulário (conjunto de palavras usuais, ou formalmente, uma coleção de símbolos terminais), um conjunto de símbolos não terminais e um conjunto de regras de produção. Sendo esse um modelo bem conhecido para geração de linguagem natural [61].

Sentenças são geradas, iniciando com um símbolo não terminal inicial, seguido da aplicação repetida de regras de transição, cada uma transformando um símbolo não terminal em uma sequência de terminais (palavras) e outros não-terminais. Esse processo é repetido até que a sequência gerada seja composta apenas por símbolos terminais.

Comparado com o modelo *n-gram*, que ignora a estrutura linguística, o modelo linguístico baseado em CFG reflete essa estrutura. A estrutura é usualmente expressa como um conjunto de regras que possuem uma simples e atômica categoria gramatical à esquerda de cada regra, e a uma sequência de categorias atômicas e palavras à direita [61]. Exemplos de regras são apresentados a seguir.

S	→	NP VP	(Sentenças consistem de uma frase substantiva seguida de uma frase verbal)
NP	→	Det N	(Frase substantiva consiste de um determinante seguido de substantivo)
NP	→	NP PP	(ou NP consiste de outra NP seguida de uma frase prepositiva)
PP	→	P NP	(PP consiste de uma preposição seguida de uma frase substantiva)
VP	→	V NP	(Frase verbal consiste de uma Preposição seguida de uma Frase Substantiva)
N	→	palavra	(Substantivo consiste de uma palavra)
...	→	...	...

Uma outra característica importante das regras, tais como as apresentadas, é a possibilidade de definir as categorias gramaticais de forma mútua e recursiva. Por exemplo, na terceira regra apresentada, NP aparece tanto do lado esquerdo quanto do lado direito da regra.

O uso da estrutura da linguagem natural é importante se a quantidade de dados de treinamento não é suficiente para estimar todos os parâmetros do modelo linguístico *n-gram*. A estrutura capturada pelo CFG permite fazer uma generalização de um conjunto de dados para outro. Contudo, o maior desafio no uso de CFG como modelo linguístico em reconhecimento de fala é a cobertura do problema, quando o CFG projetado pode não validar todas as possíveis sentenças pronunciadas pelo locutor [61]. Um outro desafio é a dificuldade da busca (decodificação) em um sistema de reconhecimento de fala que procura por uma melhor sequência de palavras que foi construída por uma gramática especificada por um CFG.

A principal extensão do CFG é o CFG probabilístico. No CFG convencional, a probabilidade de uma sentença é limitada a 1 ou 0, dependendo se ela segue as regras do CFG ou não. No caso do CFG probabilístico, a probabilidade de uma sentença é um valor contínuo, semelhante aos modelos *n-gram* [61].

Em um CFG probabilístico cada regra é associada a uma probabilidade condicional para o(s) símbolo(s) do lado direito, dado(s) os símbolos do lado esquerdo. Por exemplo, considerando a seguinte regra: NP → NP PP, à qual pode ser associada a probabilidade de 0,3. Isto significa que dada uma frase na categoria NP, existe a possibilidade de 0,3 que ela consista em outra NP seguida de uma PP (e.g., um homem com um telescópio).

Uma gramática livre de contexto probabilística (ou estocástica) coloca a distribuição probabilidade nas transições emanadas de cada símbolo não-terminal, induzindo assim uma distribuição sobre todo o conjunto de sentenças [67]. Estas probabilidades de transição podem ser estimadas de um *corpus* detalhado utilizando o algoritmo *Inside-Outside* [68], um algoritmo de maximização da estimação [69].

2.2.4 Modelagem Linguística: Máxima-Entropia

Modelo de máxima entropia, também chamado de modelo exponencial, resolve o problema de fragmentação associado a todos os modelos discutidos até agora. Fragmentação de dados

refere-se ao fato de que uma modelagem mais detalhada conduz a menos dados para estimação de novos parâmetros. Por exemplo, à medida que árvores de decisão crescem, cada folha contém menos e menos pontos de dados para estimação de parâmetros. Igualmente, à medida que N aumenta nos modelos *n-gram*, menos dados ficam disponíveis para treinamento das probabilidades.

O problema de fragmentação pode ser evitado usando um modelo exponencial na forma de [61, 67]:

$$P(w|h) = \frac{1}{z(h)} exp\left[\sum_i \lambda_i f_i(h,w)\right], \tag{2.39}$$

sendo λ_i os parâmetros do modelo, $z(h)$ o termo de normalização, e as características $f_i(h,w)$ que são funções arbitárias do histórico de palavras do par. Com base em um corpo de treinamento, a máxima probabilidade estimada [67]:

$$\sum_h P(h) \cdot \sum_w P(w|h) \cdot f_i(h,w) = E_P f_i(h,w), \tag{2.40}$$

sendo P a distribuição empírica do corpo de treinamento.

Uma das maiores limitações dos modelos exponenciais é custo computacional que, algumas vezes, os torna inviável, devido à etapa de treinamento e normalização explícita [61, 67]. Uma tentativa não normalizada é apresentada em [70] e uma suavização do modelo analisada em [71]. Alguns outros trabalhos abordam o uso de modelos exponenciais como [72, 73, 74, 75].

2.2.5 Modelagem Adaptativa

As técnicas citadas até agora tratam a linguagem como uma fonte homogênea, contudo a linguagem natural é bastante heterogênea, com tópicos, gêneros e estilos variados.

Quando os dados de teste são provenientes de uma fonte à qual o modelo linguístico não foi exposto durante o treinamento, esses dados de teste são a única informação útil para adaptação do modelo linguístico [61, 67]. Esse problema é chamado adaptação *cross-domain*. Uma técnica bastante utilizada para explorar essas informações é o histórico (continuamente desenvolvido), que é usado para criar, em tempo de execução, um *n-gram* dinâmico $P_{historico}(w|h)$, o qual, por sua vez, é interpolado com o modelo estático:

$$P_{adaptativo}(w|h) = \lambda P_{estatico}(w|h) + (1-\lambda) P_{historico}(w|h). \tag{2.41}$$

O peso no processo de interpolação é otimizado separadamente. Modelos linguísticos

adaptativos foram introduzidos por [76] e [77]. [78, 79] apresentam redução de perplexidade e [80] redução da taxa de erro de reconhecimento. [81] apresenta ainda um outro esquema de adaptação que melhora o modelo bigrama em 10% em termos de perplexidade no conjunto de teste.

No modelo de linguagem adaptativo intradomínio, os dados de teste vêm da mesma fonte dos dados de treinamento, mas estes podem ser heterogêneos, consistindo de muitos subconjuntos de dados. Estes podem consistir de variações de tópicos, estilos ou ambos. Assim, a adaptação procede da seguinte forma [61, 67]:

1. Agrupar os dados de treinamento pela principal fonte de variação, tal como um tópico.
2. Em tempo de execução, identificar o tópico ou conjunto de tópicos dos dados de teste.
3. Localizar subconjuntos apropriados do corpo de treinamento e utilizá-los para construir um modelo específico.
4. Interpolar esse modelo específico com o modelo genérico treinado utilizando todos os dados heterogêneos de treinamento.

O último passo da interpolação é importante porque o modelo específico tende a ser tendencioso embora seja menos variável. O contrário é verdadeiro para o modelo genérico. A interpolação proporciona um equilíbrio entre os dois [61, 67].

2.2.6 Modelagem da Pronúncia

O dicionário da pronúncia é a ligação entre a representação acústica e a saída dos itens léxicos do sistema de reconhecimento. A precisão dos modelos acústicos é parcialmente dependente da consistência do dicionário de pronúncia.

Associada a cada entrada léxica existe uma ou mais pronúncias descritas com base na unidade elementar escolhida (usualmente fonemas ou fones). Normalmente, são aplicados algoritmos *data-driven*, que tipicamente incluem quatro etapas [39]:

1. Geração das transcrições fonéticas por meio de um reconhecedor;
2. Alinhamento das transcrições automáticas por meio de pronúncias canônicas criadas manualmente;
3. Mapeamento das regras das pronúncias canônicas para as variantes;
4. "Enxugamento/podagem" das regras.

Uma limitação dessa abordagem é que as pronúncias canônicas de referência devem estar disponíveis. Um sistema de auto-atendimento da IBM [82] utilizou uma abordagem de geração de pronúncia baseada apenas em características acústicas [83]. A vantagem dessa

abordagem é que nenhuma pronúncia canônica é necessária, o que se torna mais prático, uma vez que, muitas palavras não existem no dicionário de pronúncia. Os experimentos realizados em [82] mostraram uma redução de 17% no erro relativo, quando o conjunto de treinamento e o conjunto de teste apresentavam sobreposição nas palavras não vistas.

Devem ser feitas considerações sobre o custo-benefício ao adicionar variações no dicionário, uma vez que, se por um lado, são gerados modelos que se assemelham mais precisamente ao sinal acústico atual, por outro lado, aumenta-se o número de modelos confusos para as palavras [64]. Diante disso, ao invés de aumentar o número de variações de pronúncia no dicionário, em [84] foi introduzido um modelo de distorção da pronúncia, para repontuar as n-melhores hipóteses geradas por uma primeira etapa de reconhecimento.

2.3 Considerações Gerais

Avanços obtidos em sistemas de reconhecimento automático de fala, baseados na modelagem acústica, têm possibilitado bons resultados no que se refere ao reconhecimento de palavras isoladas. Contudo, ainda existem certas limitações para se obter o reconhecimento automático de fala contínua e/ou espontânea. Para se atingir tal objetivo, outras técnicas têm sido utilizadas em conjunto com a modelagem acústica, como as modelagens linguística e de pronúncia.

Na modelagem linguística, são capturadas regularidades da linguagem falada, que são utilizadas para estimar a probabilidade da sequência de palavras. O método linguístico mais popular é o chamado modelo *n-gram*, que objetiva capturar as restrições sintáticas e semânticas da língua a partir da estimativa da frequência de sequência de n palavras anteriores. Outro método também muito utilizado é a modelagem linguística por meio de Árvores de Decisão, no qual é utilizada uma abordagem *top-down* de agrupamento das amostras, que estão inicialmente em único grupo, e vão sendo separadas em grupos menores, por meio de perguntas que vão sendo realizadas sobre tais amostras. Árvores de Decisão têm sido muito utilizadas na modelagem linguística para relacionar/ ligar os parâmetros do HMM, bem como, melhorar o desempenho dos modelos *n-gram*. Ainda na modelagem linguística, podem ser utilizadas as Gramáticas Livres de Contexto. Essa estrutura é usualmente expressa como um conjunto de regras que possuem uma simples e atômica categoria gramatical à esquerda de cada regra, e a uma sequência de categorias atômicas e palavras à direita. Neste trabalho, também foram apresentadas as técnicas de Máxima-Entropia e de Modelagem Adaptativa. A primeira visa resolver o problema de fragmentação dos modelos anteriormente discutidos, e a segunda trata a natureza heterogênea da fala natural, que pode variar de tópicos, gêneros e estilos.

A modelagem acústica tem como principal objetivo lidar/tratar com a variabilidade presente em um sinal de fala. Essa variabilidade pode advir do contexto linguístico, ou pode estar associada ao contexto não-linguistico, como por exemplo, o locutor (e.g. características

físicas, forma da elocução, humor, etc.), ou ambiente acústico (e.g. ruído de fundo), além do canal de gravação (e.g. microfone, telefone, etc.). Nessa modelagem, predominam os métodos de análise espectral LPC (*Linear Predictive Coding*) e o método de análise espectral por banco de filtros, os quais também foram descritos neste capítulo.

A modelagem da pronúncia é a ligação entre a representação acústica e a saída dos itens léxicos do sistema de reconhecimento. Associada a cada entrada léxica existe uma ou mais pronúncias descritas com base na unidade elementar escolhida (usualmente fonemas ou fones). Devem ser feitas considerações sobre o custo-benefício ao adicionar variações no dicionário, uma vez que, se por um lado, são gerados modelos que se assemelham mais precisamente ao sinal acústico atual, por outro lado, aumenta-se o número de modelos confusos para as palavras.

Apesar de exercerem papéis diferentes no processo de reconhecimento, na etapa de classificação dos padrões gerados, as modelagens acústica, linguística e de pronúncia têm como fundamento básico a teoria de decisão de Bayes, que considera a probabilidade de uma determinada sentença a partir de um espaço de observação. Na etapa de classificação, o método que tem apresentado melhores resultados é o HMM, cujo funcionamento será explicado no Capítulo 3.

Capítulo 3

Modelos Ocultos de Markov Aplicados ao Reconhecimento da Fala

De uma forma geral, os métodos conhecidos para reconhecimento de fala diferenciam-se quanto à forma como os parâmetros extraídos são utilizados na construção e classificação dos padrões. Dessa forma, podem ser divididos em dois grupos [85]: Métodos Paramétricos e Métodos Estatísticos.

Nos métodos paramétricos, é levada a efeito uma redução de dados explícita, após a qual é obtido um padrão de referência que continua ainda na forma paramétrica. A regra de decisão no processo de comparação de padrões baseia-se em medidas de distância. Como exemplo de método paramétrico, pode-se citar a Quantização Vetorial.

Nos métodos estatísticos, a construção dos padrões é obtida por meio de modelos estatísticos, tais como Modelos Ocultos de Markov (HMMs) [86, 87]. Os parâmetros extraídos são, portanto, com o auxílio da teoria das probabilidades, representados por modelos estocásticos nos quais está presente uma redução implícita de dados. Nesses métodos não é feita uma comparação direta de padrões e a decisão é tomada usando o cálculo de probabilidades associadas aos modelos.

Muitos sistemas aplicados ao reconhecimento de fala para vocabulário grande e contínuo utiliza os Modelos Ocultos de Markov (*Hidden Markov Models* – HMM) para classificação na modelagem acústica [2, 8, 9, 20, 38, 39, 86, 88]. Esse método consiste na modelagem da função densidade probabilidade de uma sequência de vetores de características acústicas.

Para estimar os parâmetros do HMM existem algoritmos eficientes. Em reconhecimento de fala, os HMM podem ser usados para representar palavras, sentenças ou unidades menores tais como fones [89, 90].

3.1 Definição e descrição do modelo

Em um HMM, formado por uma cadeia de estados (cadeia de Markov), existem dois processos estocásticos associados, um envolvendo as transições entre os estados e outro envolvendo

as observações de saída de cada estado.

As observações de saída são manifestações do fenômeno sendo modelado, e são descritas por funções probabilísticas que podem ser obtidas de duas formas. A primeira delas, usualmente utilizada no modelamento acústico do sinal de fala, está associada à emissão de um símbolo no instante de chegada a um estado, e é conhecida como máquina de Moore. A segunda forma, geralmente utilizada no processamento de linguagem, está associada à emissão de um símbolo durante a transição entre estados, e é conhecida como máquina de Mealy [46].

Na Figura 3.1 é apresentado um exemplo de modelo HMM de 3 estados, correspondente às formas de Moore e Mealy, respectivamente, em que:

i, j e k	– representam os estados do modelo;
a_{ij}	– representa a probabilidade de transição do estado i para o estado j;
b	– representa a probabilidade de emissão de um símbolo, associado ou a um estado (b_i) ou a uma transição entre estados (b_{ij}).

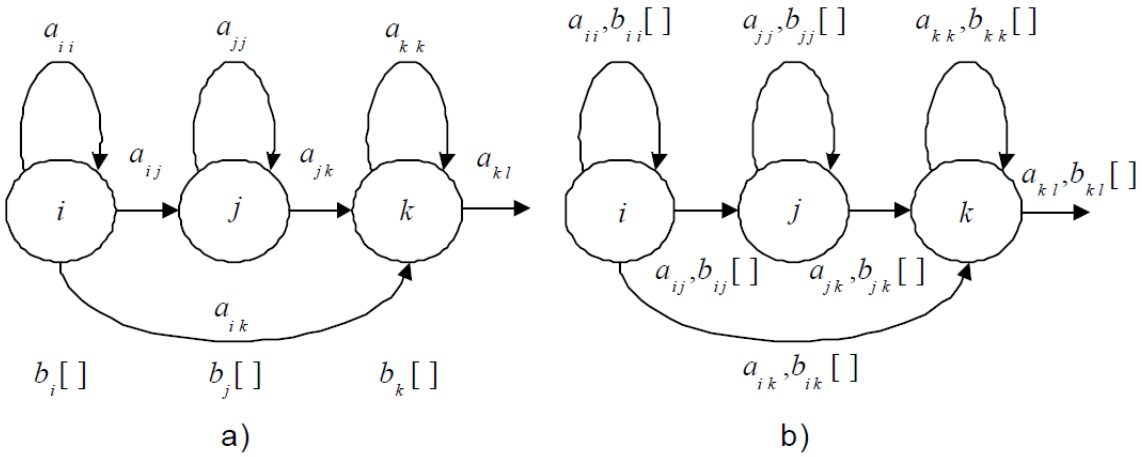


Figura 3.1: *Exemplo de um HMM de 3 estados, associado à topologia Moore (a) e Mealy (b).*

Em geral, para o reconhecimento de fala, utiliza-se um modelo simplificado de HMM conhecido como modelo *left-right* (esquerda-direita), ou modelo de Bakis [13]. Neste modelo, exemplificado na Figura 3.2, são permitidos apenas transições para o mesmo estado, ou transições de um estado i para um estado j, mais à direita, sendo $a_{ij} = 0$ se $j > i + 2$ ou $j < i$.

Quando se aplica HMM a problemas de reconhecimento, as observações de saída são definidas por parâmetros representativos do sinal de fala, tais como os MFCC (*Mel Frequency Cepstral Coefficients*), LFCC (*Linear Frequency Cepstral Coefficients*), LPC (*Linear Predictive Coding Coefficients*), entre outros.

Um HMM pode ser definido como sendo:

- Um conjunto de estados S_j, incluindo um estado inicial S_i e um estado final S_f;

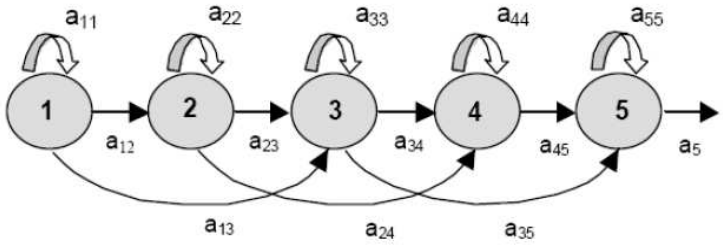


Figura 3.2: *Exemplo de um HMM tipo left-right de cinco estados.*

- Uma matriz de transições $A = a_{ij}$, sendo a_{ij} representa a probabilidade de se efetuar uma transição do estado i para o estado j;
- Uma matriz de probabilidades de saída $B = b_j(k)$, sendo $b_j(k)$ define a probabilidade de emissão do símbolo k, ao se chegar ao estado j (modelo de Moore). O símbolo k pertence a um conjunto finito ou infinito de símbolos de saída.

Desde que a_{ij} e $b_j(k)$ sejam probabilísticos, as seguintes propriedades devem ser satisfeitas:

$$\sum_j a_{ij} = 1 \quad \forall i, j \quad , sendo \quad a_{ij} \geq 0. \tag{3.1}$$

$$\sum_k b_j(k) = 1 \quad \forall k \quad , sendo \quad b_j(k) \geq 0. \tag{3.2}$$

Pode-se associar a sequência de transições de estado a um processo estocástico X_t, e a sequência de símbolos emitidos a outro processo Y_t , dependente do primeiro. Assim, a cada incremento em t existirá exatamente uma transição de estado (mesmo que seja para o mesmo estado – autotransição) e, consequentemente, a emissão de um símbolo. Utilizando esta notação temos:

$$a_{ij} = P(X_t = j | X_{t-1} = i), \tag{3.3}$$

$$b_j(k) = P(Y_t = k | X_t = j), \tag{3.4}$$

sendo $X_t = j$ significa que a cadeia de Markov está no estado j no instante t, e $Y_t = k$ significa que o símbolo de saída emitido no instante t é k.

O conjunto de símbolos observáveis k corresponde à saída física do sistema que está sendo modelado. Desta forma, para sistemas de reconhecimento de fala, os símbolos devem corresponder a um conjunto de vetores de parâmetros extraídos do sinal sendo analisado.

É comum o uso da notação observada em (3.5), utilizada para indicar um modelo HMM [91]:

$$\lambda = (A, B, \pi), \tag{3.5}$$

sendo:

- A é a matriz de elementos $a_{ij} = P[S_j|S_i]$, chamada matriz de transição de estados, de dimensão N^2, sendo N definido como o número de estados. S_i e S_j representam dois estados quaisquer do modelo;
- B é a matriz de elementos $b_j(k) = P[V_k|S_j]$, que representa a probabilidade de se observar o resultado V_k (conjunto de resultados de saída esperados do modelo) estando no estado S_j. De acordo com a natureza dos elementos da matriz, os modelos HMM podem ser classificados como discretos, contínuos ou semi-contínuos;
- π: vetor de probabilidade inicial $\pi(i)$. No caso do modelo *left-right*, $\pi(i) = 1$ para $i = 1$ e $\pi(i) = 0$ para $i > 1$.

3.2 Tipos de HMM

Dependendo do tipo de função de probabilidade escolhida para os símbolos de saída, o HMM pode ser classificado como discreto, contínuo ou semi-contínuo.

1. **Discreto:** O HMM é dito discreto quando o número de possíveis símbolos de saída, K, é finito e a probabilidade de se emitir o símbolo V_k , no estado S_j , é dada por $b_j(k)$.

   A função de probabilidade B, mencionada em (3.5), tem as seguintes propriedades:

$$b_j(k) \geq 0, \quad 1 \leq j \leq N \quad e \quad 1 \leq k \leq N; \tag{3.6}$$

$$\sum_{k=1}^{K} b_j(k) = 1, \quad 1 \leq j \leq N; \tag{3.7}$$

   sendo:

   - $b_j(k) = P[V_k|S_j]$
   - N - representa o número de estados do modelo;
   - K - representa o número de símbolos de saída.

No modelo HMM discreto, as sequências de observações são formadas por índices de vetores de um dicionário. Este dicionário é formado por um conjunto de vetores chamados de palavras-código ou vetores-código. Em cada quadro do sinal de fala obtém-se um vetor de parâmetros que, após a quantização vetorial, é associado a um dos K possíveis vetores-código.

2. **Contínuo:** O HMM é dito contínuo quando sua função densidade de probabilidade for contínua. Usualmente, utiliza-se a função densidade de probabilidade modelada como uma mistura finita de M gaussianas multidimensionais, dada por:

$$b_j(O_t) = \sum_{m=1}^{M} c_{jm} G(O_t, \mu_{jm}, U_{jm}), \quad 1 \leq j \leq N; \tag{3.8}$$

sendo:

- O_t : é o vetor de parâmetros de entrada, suposto de dimensão D, no instante t;
- c_{jm} : é o coeficiente da $m-$ésima componente da mistura no estado S_j;
- G : é uma função densidade de probabilidade gaussiana multidimensional (dimensão D) com vetor média m_{jm} e matriz covariância U_{jm};
- M - representa o número de mistura gaussiana;
- N - representa o número de estados.

Os coeficientes c_{jm} e a função densidade de probabilidade da mistura devem satisfazer as seguintes definições:

$$\sum_{m=1}^{M} c_{jm} = 1, \quad 1 \leq j \leq N; \tag{3.9}$$

$$c_{jm} \geq 0, \quad 1 \leq j \leq N; \tag{3.10}$$

$$\int_{-\infty}^{+\infty} b_j(x) dx = 1. \tag{3.11}$$

Sendo a integral da Equação 3.11 multidimensional (vetor x de dimensão D). No modelo HMM contínuo a matriz B é representada pelo conjunto $B = b_j(.)$ de funções densidade de probabilidade, indexadas nos estados do HMM.

Quando as misturas (vetor média μ_{jm} e matriz covariância U_{jm}) são iguais para todos os estados do modelo, o HMM contínuo é denominado HMM "tied" contínuo [85]. Neste

caso, de um estado para outro, haverá somente variação dos coeficientes de ponderação (c_{jm}) das componentes da mistura.

3. **Semi-contínuo:** O HMM é dito semi-contínuo quando o modelo for um caso intermediário entre os HMM discreto e contínuo. Neste caso, a densidade de probabilidade de emissão dos símbolos de saída é dada por:

$$b_j(O_t) = \sum_{V_k \in \eta(O_t)} c_j(k) f(O_t|V_k), \quad 1 \leq j \leq N, \tag{3.12}$$

sendo:

- O_t o vetor de entrada;
- K o número de funções densidade de probabilidade (símbolos de saída);
- $\eta(O_t)$ o conjunto das funções densidade de probabilidade que apresentam os M maiores valores de $f(O_t|V_k)$, sendo $1 \leq M \leq K$;
- V_k o $k-$ésimo símbolo de saída;
- $c_j(k)$ a probabilidade de emissão do símbolo V_k no estado S_j;
- $f(O_t|V_k)$ o valor da $k-$ésima função densidade de probabilidade;
- N representa o número de estados utilizados.

No HMM semi-contínuo, o conjunto de funções densidade de probabilidade é o mesmo para todos os estados e todos os modelos, havendo modificação apenas dos coeficientes de ponderação $c_j(k)$.

Os coeficientes $c_j(k)$, os parâmetros das funções densidade de probabilidade e as probabilidades de transição são calculados e otimizados conjuntamente. Quando o valor de M for igual a K, pode-se considerar o HMM semi-contínuo como um HMM contínuo, em que todas as misturas são iguais para todos os estados e todos os modelos, variando-se apenas os valores dos coeficientes das misturas de um estado para outro [85].

3.3 HMM em reconhecimento de fala

O HMM pode ser utilizado na representação de quaisquer unidades ou subunidades da fala, tais como palavras, fones, difones, trifones, etc [17, 85, 92, 93]. Quando são criados modelos para as subunidades, a representação da palavra desejada é construída concatenando-se os modelos das subunidades correspondentes. Um exemplo de subunidade bastante utilizada são os fones, que podem ser classificados como dependentes de contexto ou independentes

de contexto. Nos fones independentes de contexto, os modelos HMM são treinados independentemente, levando em consideração apenas a sua ocorrência e não o contexto em que ocorrem[46].

Uma das vantagens do uso da palavra como unidade está no fato da melhor representação dos efeitos da coarticulação. Porém, a escolha da unidade básica a ser modelada deve ser feita com cuidado, levando-se em consideração a viabilidade da obtenção dos dados para o treinamento desta unidade. Além disso, à medida que o tamanho da unidade básica aumenta, o número de unidades utilizadas na obtenção de qualquer palavra cresce exponencialmente, juntamente com a quantidade de dados necessária para o treinamento do sistema[85].

Uma alternativa bastante razoável para escolha da unidade básica, é a utilização de subunidades fonéticas, que necessitam de vários exemplos de cada subunidade, e não mais de vários exemplos de cada palavra. Essa escolha pode ser considerada uma vantagem quando são utilizados vocabulários grandes, em que a disponibilidade de um grande número de exemplos para cada palavra torna-se inviável.

Na escolha do número de estados ótimo, para representação do modelo HMM, não há regras. A maneira mais comum consiste em fazer o número de estados igual ao número de fonemas da palavra. Nesse caso, os modelos que representam palavras diferentes apresentarão números de estados diferentes [85].

Quanto ao número de símbolos de saída (HMM discreto), ou ao número de gaussianas por mistura (HMM contínuo) a ser utilizado, deve-se ter o compromisso entre o tamanho da sequência de treinamento e o número de parâmetros a estimar, de modo que se obtenha boas estimativas desses.

Os sistemas de reconhecimento de fala geralmente são considerados como modelos de Markov de primeira ordem, para os quais são feitas duas hipóteses [46]:

- **Hipótese de Markov:** A probabilidade de uma cadeia estar em um dado estado, no instante t, depende apenas de seu estado no instante $t-1$;
- **Hipótese de Independência de Emissão de Símbolos:** A probabilidade de emissão de um símbolo de saída, no instante t, depende apenas da transição realizada neste instante de tempo.

Os modelos de Markov, na prática, são usados para representar uma sequência de observações. No reconhecimento de fala, deseja-se determinar qual será o modelo λi (entre vários modelos disponíveis) que melhor representará a sequência de observações desejada:

$$max_i[P(\lambda_i|O)] = max_i[P(O|\lambda_i)].P(\lambda_i). \tag{3.13}$$

Assumindo que os modelos λ_i são equiprováveis, a Equação 3.13 resulta:

$$max_i[P(\lambda_i|O)] = max_i[P(O|\lambda_i)]. \tag{3.14}$$

A estimação da Equação 3.14 consiste na própria identificação da palavra pronunciada, que é essencialmente um procedimento de classificação, em que deseja-se classificar uma determinada sequência de observação O em alguma categoria i. Em outras palavras, deseja-se determinar o modelo λ_i que apresente a maior probabilidade de reproduzir a sequência de observação desejada (O).

Nos sistemas utilizando HMM discreto, cada vetor de parâmetros $\hat{O}_i$ é representado por um símbolo pertencente a um conjunto de K símbolos, tornando-se útil a utilização da quantização vetorial. Dessa forma, tem-se uma densidade de probabilidade discreta associada a cada estado dos modelos.

Nos modelos utilizando HMM contínuo, por outro lado, não se emprega a quantização vetorial. A densidade de probabilidade associada a cada estado dos modelos é contínua, sendo representada normalmente por uma mistura de gaussianas multidimensionais. Nesse caso, evita-se o erro de quantização gerado na quantização vetorial dos vetores de parâmetros acústicos mas, em compensação, a estimação do modelo é bem mais complexa e requer uma quantidade maior de dados para treinamento [46].

3.4 Modelagem do HMM

Dado um HMM, existem três questões básicas que devem ser resolvidas: a avaliação, a decodificação e o treinamento.

No treinamento, deseja-se determinar os parâmetros do modelo que maximizem a probabilidade de geração da observação. Neste procedimento utiliza-se como solução o algoritmo *Forward-Backward*, também conhecido como algoritmo de re-estimação de *Baum-Welch* [85].

Na avaliação deseja-se determinar qual o modelo, dentre os vários modelos, que mais provavelmente gerou uma dada sequência de observações. Nesse procedimento, utiliza-se como solução o algoritmo *Forward* ou o algoritmo de [85].

Na decodificação, utiliza-se como solução o algoritmo de Viterbi que, a partir de uma sequência de observações, tem como função determinar a sequência de estados que mais provavelmente produziu as observações.

3.4.1 Solução do problema de treinamento

Na fase de treinamento, é feita a estimação dos parâmetros dos modelos, $\lambda_l = (A, B, \pi)$ um modelo para cada $l-$ésima unidade de treinamento (e.g. palavra, fone, trifone). Desde que exista um procedimento de re-estimação convergente para o modelo de densidades discretas, teoricamente é possível escolher-se aleatoriamente valores iniciais para cada um dos

parâmetros do modelo (sujeitos às restrições iniciais) e deixar a re-estimação determinar os valores ótimos (máxima verossimilhança), que corresponderão aos HMM de referência, um para cada uma das L unidades de treinamento.

A estimação pode ser realizada usando o processo iterativo de *Baum-Welch*, que pode ser descrito por meio dos passos [87]:

1. Atribuição inicial dos valores para os parâmetros do modelo $\lambda_l = (A, B, \pi)$ e para a probabilidade P_l;

2. Re-estimação dos parâmetros do modelo com base no algoritmo de re-estimação de *Baum-Welch* (as equações serão descritas a seguir), obtendo-se $\bar{\lambda}_l$;

3. Cálculo da probabilidade $\bar{P}_l$ associada ao modelo $\bar{\lambda}_l$ re-estimado e comparação com a probabilidade anteriormente calculada P_l;

4. Se $\bar{P}_l - P_l \leq \delta$ (limiar), o processo de re-estimação é finalizado. Caso contrário, retorna-se ao passo 2.

As atribuições iniciais dos parâmetros do modelo discreto devem obedecer regras simples, de forma a satisfazer as restrições do modelo "esquerda-direita". O vetor de probabilidade inicial é $\pi_i = 1, \ldots, 0$, visto que o modelo é "esquerda-direita"e, portanto, sempre é inicializado no estado 1, não sendo necessário re-estimá-lo. A matriz $A = [a_{ij}]$ inicial é gerada obedecendo a seguinte restrição: $a_{ij} = 0, j < i, j > i + 2$, já que para modelos "esquerda-direita"um estado visitado no instante de tempo t não poderá ser visitado em um instante posterior. Esta restrição deverá se manter até o final do processo de re-estimação. Para a matriz $B = [b_i(k)]$, assume-se que todos os símbolos nos estados são "igualmente prováveis"e $b_j(k)$ inicia-se com $1/M$ para todo j, k, para simplificar.

As equações do método de re-estimação de *Baum-Welch* são [87]:

1. $\overline{a_{ij}}$ = (número esperado de transições do estado q_i para o estado q_j) / (número esperado de transições do estado q_i).

2. $\overline{b_j(k)}$ = (número esperado de vezes no estado j observando o símbolo w_k)/(número esperado de vezes no estado j).

ou seja,

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i) \beta_t(i)} \quad , 1 \leq i \leq N \quad , 1 \leq j \leq N, \tag{3.15}$$

$$\overline{b_j(k)} = \frac{\sum_{t=1, O_t = w_k}^{T} \alpha_t(i) \beta_t(j)}{\sum_{t=1}^{T} \alpha_t(j) \beta_t(j)} \quad , 1 \leq j \leq N \quad , 1 \leq k \leq M, \tag{3.16}$$

com:

$$\sum_{j=1}^{N} a_{ij} = 1; \sum_{k=1}^{M} b_j(k) = 1; \sum_{i=1}^{N} \pi_i = 1, \quad a_{ij} \geq 0; b_j(k) \geq 0; \pi_i \geq 0. \tag{3.17}$$

Cada parâmetro $b_j(O_t), 1 \leq j \leq N$ e $1 \leq t \leq T$, é obtido a partir da comparação (em relação a um dado estado j e variando t), com os valores da matriz $[b_j(k)]$ referentes ao índice k do símbolo associado ao vetor O_t no mesmo estado j. Atribui-se a $b_j(O_t)$ o valor de $b_j(k)$ correspondente ao referido símbolo w_k, no estado j.

A probabilidade $\alpha_t(i)$ é denominada probabilidade de avanço (*forward probability*), pois está associada à ocorrência de uma dada sequência de observações $O^l = O_1, O_2, \ldots, O_T$, segundo o tempo crescente (iniciando em $t = 1$ indo até $t = T$), sendo formulada em [87] como:

1. Inicialização:

$$\alpha_i(i) = \pi_i b_i(O_i) \quad , 1 \leq i \leq N; \tag{3.18}$$

2. Indução:

$$\alpha_{t+1}(j) = \left\{ \sum_{i=1}^{N} \alpha_t(i) a_{ij} \right\} b_j(O_{t+1}), \quad 1 \leq t \leq T-1, \quad 1 \leq j \leq N. \tag{3.19}$$

A probabilidade P_l, associada ao modelo $\lambda_l = (A, B, \pi)$, é determinada por [87]:

$$P_l = Prob(O^l|\lambda_l) = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j), \tag{3.20}$$

para algum t, $l \leq t \leq T$.

Fazendo-se $t = T - 1$, obtém-se:

$$P_l(O^l|\lambda_l) = \sum_{i=1}^{N} \alpha_T(i), \tag{3.21}$$

sendo

$$\alpha_T(i) = P_l(O_1, \ldots, O_T, q_T = i|\lambda_l). \tag{3.22}$$

O cálculo das probabilidades de avanço (*forward*), inicia-se atribuindo-se ao estado q_i o vetor inicial O_1. O passo de indução é o ponto principal do cálculo da probabilidade de avanço, como ilustrado na Figura 3.3.

Na Figura 3.3, é exibido como estado q_j pode ser alcançado no instante de tempo $t+1$ a partir dos N possíveis estados, q_i, $1 \leq i \leq N$, no instante de tempo t associado à sequência de

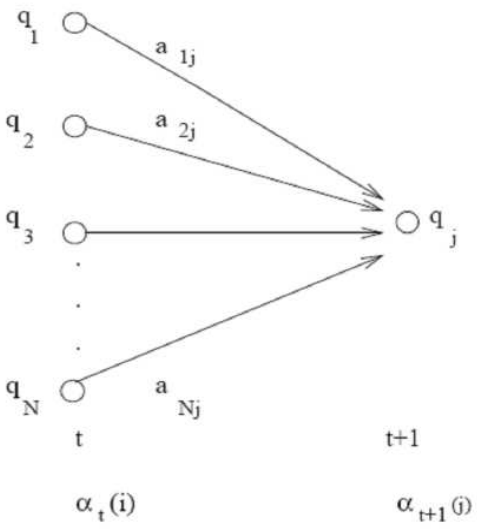


Figura 3.3: *Ilustração da sequência de operações necessárias à computação da variável forward* $\alpha_{t+1}(j)$.

vetores O^l. Assim $\alpha_t(i)$ é a probabilidade de que os vetores $O_1, O_2, \ldots, O_T$ tenham ocorrido estando no estado q_i no instante t. O produto $\alpha_t(i)a_{ij}$ é então a probabilidade de que o evento $O_1, O_2, \ldots, O_T$ seja observado a partir de q_i no instante t, tal que o estado q_j seja alcançado no instante $t+1$. Somando esse produto ao longo dos N estados possíveis q_i, $1 \leq i \leq N$ no instante t, obtém-se a probabilidade associada ao estado q_j no instante $t+1$. Desde que isso seja feito e q_j seja conhecido, é fácil ver que $\alpha_{t+1}(j)$ é obtido de acordo com o vetor O_{t+1}, no estado j, ou seja, multiplicando as quantidades somadas pela probabilidade $\alpha_{t+1}(j)$. A computação da Equação 3.21 é realizada para todos os $j-$ésimos estados, $1 \leq j \leq N$, para um dado t; a computação é, então, iterativa para $t = 1, 2, ..., T-1$.

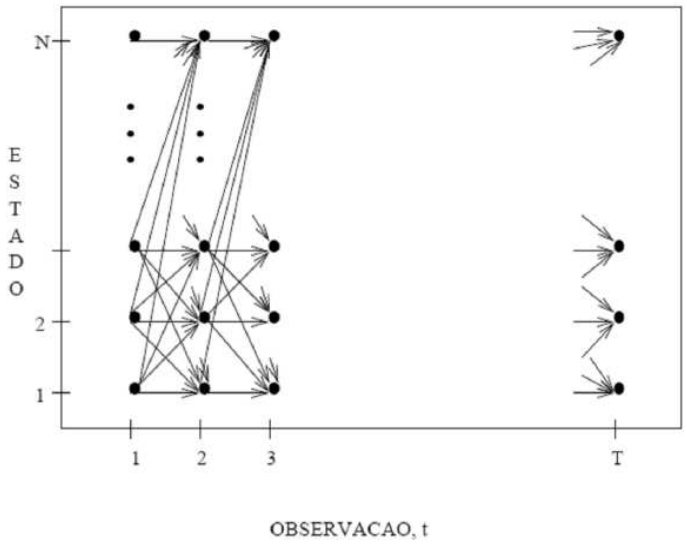


Figura 3.4: *Implementação da computação de* $\alpha_t(i)$ *em termos de uma treliça de observações* t *e estados* i.

O cálculo da probabilidade de avanço é baseado na estrutura treliça mostrada na Fi-

gura 3.4. Desde que exista somente N estados (nós para cada instante de tempo na treliça), todas as possíveis sequências de estado serão agrupadas dentro desses N nós, não importando o tamanho da sequência de observações. No instante $t = 1$, o primeiro instante de tempo na treliça, referente ao primeiro vetor O_1 de uma dada sequência de observações O^l, é necessário calcular valores de $\alpha_1(i), 1 \leq i \leq N$. Para os instantes $t = 2, 3, \ldots, T$, é necessário calcular os valores de $\alpha_t(j), 1 \leq j \leq N$, em que cada um dos cálculos envolve somente N valores anteriores de $\alpha_{t-1}(j)$, uma vez que cada um dos N pontos da grade é obtido a partir dos mesmo N pontos da grade no instante de tempo anterior [87].

De forma similar, $\beta_t(i)$ é denominada probabilidade de retrocesso (*backward probability*), pois está associada à ocorrência da sequência de observações $O^l = O_1, O_2, \ldots, O_T$ segundo o tempo decrescente, sendo definida como [87]:

$$\beta_t(i) = P_l(O_{t+1}, O_{t+2}, \ldots, O_T | q_t = i | \lambda_l), \tag{3.23}$$

ou, seja, a probabilidade da sequência de observações parcial do instante de tempo $t+1$ até o fim, dado o estado q_i no instante de tempo t e o modelo λ_l. Assim, pode-se obter $\beta_t(i)$ indutivamente da seguinte forma [87]:

1. Inicialização:

$$\beta_T(i) = 1 \quad , 1 \leq i \leq N. \tag{3.24}$$

2. Indução

$$\beta_T(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \quad , t = T-1, T-2, \ldots, 1 \quad , 1 \leq i \leq N. \tag{3.25}$$

O passo 1, inicialização, define arbitrariamente $\beta_T(i) = 1$ para todo i. O passo 2, ilustrado na Figura 3.5, mostra que para ter ocorrido o estado q_i no instante de tempo t, levando-se em conta a sequência de observações no instante de tempo $t + 1$, é necessário considerar todos os possíveis estados q_j no instante $t + 1$, considerando a transição de q_i para q_j (o termo a_{ij}), como também a observação O_{t+1} no estado j (O termo $b_j(O_{t+1})$).

Não há nenhuma técnica iterativa ótima para re-estimar os parâmetros do modelo $A, Be\pi$, os quais maximizam $P_l(O^l|\lambda_l)$, dada uma sequência de observações finita como dados de treinamento. Entretanto, no método iterativo proposto por Baum e Welch em [86] escolhe-se λ_l tal que $P_l(O^l|\lambda_l)$ seja localmente máxima. O modelo re-estimado $\bar{\lambda}_l = (\bar{A}, \bar{B}, \pi)$ (em modelos esquerda-direita π não precisa ser re-estimado) é melhor ou igual ao modelo estimado anteriormente λ_l, desde que $P_l(O^l|\bar{\lambda}_l) \geq P_l(O^l|\lambda_l)$. Assim, utiliza-se $\bar{\lambda}_l$ no lugar de λ_l repetindo o processo de re-estimação para uma dada sequência observada, O^l , até que algum ponto limite é atingido, ou seja, é atingido um número de iterações desejado ou o valor de probabilidade escolhido.

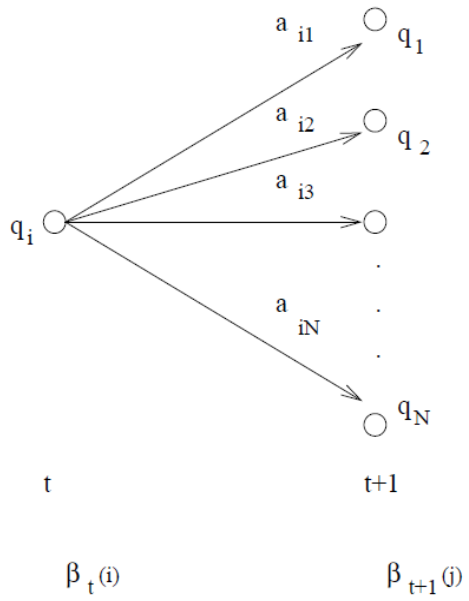


Figura 3.5: *Ilustração da sequência de operações necessárias à computação da variável backward* $\beta_t(i)$.

O resultado final ou estimado é denominado estimação de máxima verossimilhança do HMM [87], obtendo-se assim os HMM de referência, um para cada uma das L unidades de treinamento.

Uso de Múltiplas Sequências de Observações

O maior problema associado ao HMM do tipo "esquerda-direita"reside no fato de que não se pode usar uma única sequência de observações para treinar o modelo (isto é, para a re-estimação dos parâmetros do modelo) [5, 50, 87]. Isso se deve à natureza transitória dos estados dentro do modelo, permitindo apenas um pequeno número de observações para qualquer estado (até que uma transição seja feita para um estado sucessor). Assim, para obtenção de dados suficientes para se fazer estimativas confiáveis de todos os parâmetros do modelo, deve-se usar múltiplas sequências de observações.

A modificação do método de re-estimação é direta e apresentada a seguir [87].

Seja o conjunto de U sequências de observações representado por:

$$O = [O^{(1)}, O^{(2)}, \ldots, O^{(u)}]. \tag{3.26}$$

Assume-se que as sequências de observações são independentes e o objetivo é o ajuste dos parâmetros do modelo que maximizam a expressão:

$$P(O|\lambda) = \prod_{u=1}^{U} P(O^{(u)}|\lambda) = \prod_{u=1}^{U} P_u. \tag{3.27}$$

Uma vez que as fórmulas de re-estimação são baseadas em frequências de ocorrências de eventos, para as múltiplas sequências de observações essas fórmulas são modificadas adicionando-se as frequências de ocorrências individuais de cada sequência. Assim, as fórmulas de re-estimação modificadas são [87]:

$$\overline{a_{ij}} = \frac{\sum_{u=1}^{U} \frac{1}{P_u} \sum_{t=1}^{T_u-1} \alpha_t^u(i) a_{ij} b_j(O_{t+1}^{(u)}) \beta_{t+1}^u(j)}{\sum_{u=1}^{U} \frac{1}{P_u} \sum_{t=1}^{T_u-1} \alpha_t^u(i) \beta_t^u(i)} \quad , 1 \leq i \leq N, \quad 1 \leq j \leq N; \tag{3.28}$$

$$\overline{b_j(k)} = \frac{\sum_{u=1}^{U} \frac{1}{P_u} \sum_{t=1, s.t. O_t = w_k}^{T_u} \alpha_t^u(j) \beta_t^u(j)}{\sum_{u=1}^{U} \frac{1}{P_u} \sum_{t=1}^{T_u} \alpha_t^u(j) \beta_t^u(j)} \quad , 1 \leq j \leq N, \quad 1 \leq k \leq N. \tag{3.29}$$

3.4.2 Solução do problema de avaliação

Na fase de reconhecimento (identificação), é realizada a estimação da probabilidade de ocorrência de uma dada sequência de observações $O^l = O_1, O_2, \ldots, O_T$, associada a cada modelo $\lambda_l = (A, B, \pi)$, obtido durante a fase de treinamento ($1 \leq l \leq L$).

Uma vez que os HMM tenham sido treinados para cada palavra, a estratégia de identificação é direta [5, 94, 95]. Para a palavra a ser identificada, é obtida a sequência de observações $O^l = O_1, O_2, \ldots, O_T$ e gerada a tabela de códigos associadas à sequência, a partir da quantização vetorial. Em seguida, é calculada a probabilidade associada a cada modelo de referência $\lambda_l = (A, B, \pi)$ (obtido durante a fase de treinamento). Após o cálculo da probabilidade, a partir de uma regra de decisão, a palavra ou sentença é aceita ou rejeitada pelo sistema.

O procedimento para cálculo da probabilidade $P(O^l|\lambda_l)$ é o mesmo já apresentado anteriormente, descrito a seguir.

Fazendo-se $t = T - 1$, obtém-se [94]

$$P(O^l|\lambda_l) = \sum_{i=1}^{N} \alpha_T(i), \tag{3.30}$$

sendo:

$$\alpha_1(i) = \pi_i b_i(O_1) \quad , 1 \leq i \leq N, \tag{3.31}$$

$$\alpha_{t+1}(j) = \left\{ \sum_{i=1}^{N} \alpha_t(i) a_{ij} \right\} b_j(O_{t+1}) \quad , 1 \leq t \leq T-1 \quad , 1 \leq j \leq N. \tag{3.32}$$

Os coeficientes a_{ij} e π correspondem, exatamente, aos valores de referência da matriz A e vetor π, respectivamente.

Os coeficientes $b_j(O_t)$ são obtidos a partir da matriz $B = [b_j(k)]$, da seguinte forma: a

cada vetor O_t de uma $l-$ésima palavra corresponde, após a quantização vetorial, um índice do quantizador vetorial (símbolo w_k). Cada coeficiente $b_j(k)$ representa a probabilidade de ocorrência de um dado símbolo w_k, no estado j. Assim, cada coeficiente $b_j(O_t)$ corresponde ao valor da probabilidade do símbolo associado àquele estado j.

A palavra que apresentar o maior valor de probabilidade é a palavra identificada pelo sistema (ou aceita), desde que esta seja maior que um dado limiar, caso contrário, a palavra é rejeitada.

3.4.3 Solução do problema de decodificação

Diferentemente do que acontece no problema de avaliação, para o qual existe uma solução exata, existem várias maneiras possíveis de se resolver o problema de decodificação. A dificuldade de se encontrar a sequência de estados ótima, associada com uma dada sequência de observações, está exatamente na definição do que seja essa sequência ótima, isto é, existem vários critérios de otimização. Existe uma técnica formal para encontrar a sequência de estados ótima única, baseada em métodos de programação dinâmica, chamada de Algoritmo de Viterbi [5, 87, 96]. Esse algoritmo é uma solução ótima recursiva ao problema de estimar a sequência de estados de um processo de Markov discreto no tempo [5, 50].

Considerando a estrutura de treliça apresentada na Figura 3.4, a propriedade mais importante, inerente a essa estrutura, é que para cada sequência de estados possível, Q, corresponde um único caminho por meio da treliça e vice-versa [5, 50].

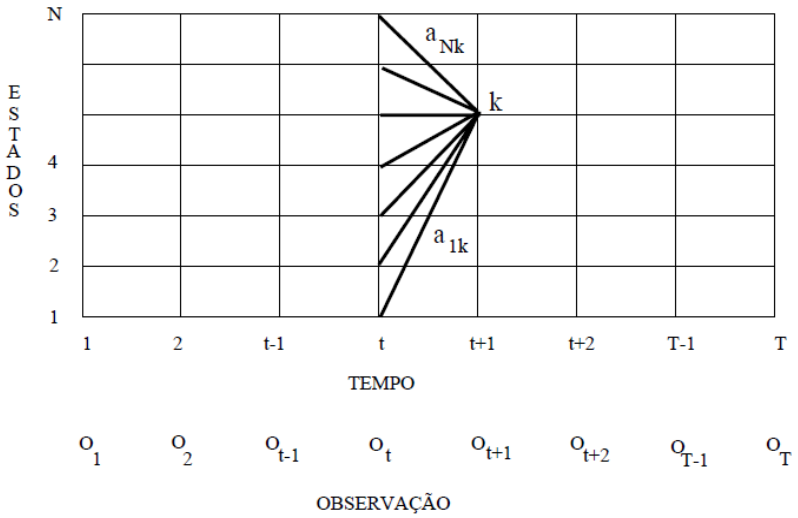


Figura 3.6: *Algoritmo de Viterbi [5].*

Observando a Figura 3.6, pode-se notar que para vários instantes de tempo diferentes, existe mais de um caminho parcial chegando em cada nó (estado), cada um com determinado comprimento (valor de probabilidade). O segmento de caminho mais curto, ou seja, aquele que apresenta maior valor de probabilidade, é chamado de "sobrevivente" correspondente a cada nó. Em outras palavras, para cada instante de tempo, existe um número de sobreviventes igual ao número de nós na treliça.

No último instante de tempo deve existir apenas um único sobrevivente, pois a cadeia de markov deve terminar em um estado bem determinado. Nesse ponto, o caminho total (de $t = 1$ até $t = T$) representa o menor caminho percorrido, ou seja, apresenta o maior valor de probabilidade. Percorrendo de volta a sequência de estados desse caminho, determina-se a sequência de estados associada que fornece o caminho mais provável, ou seja, a sequência de estados ótima.

Definindo-se a variável $\delta_t(i)$ como o maior valor de probabilidade ao longo de um único caminho até o instante de tempo t, ou seja, considerando-se as t primeiras observações que terminam no estado q_i, tem-se por indução que

$$\delta_{t+1}(j) = [\underset{i}{max}\delta_t(i)a_{ij}]b_j(O_{t+1}) \quad , 1 \leq i \leq N. \tag{3.33}$$

Para se obter a sequência de estados, é necessário reter a trilha do argumento que maximiza a Equação 5.5, para cada t e j. Para tanto, define-se a variável $\psi_t(j)$. O método para se encontrar a sequência de estados ótima é dado por [50, 87]:

1. Inicialização:

$$\delta_1(i) = \pi_i b_i(O_1) \quad , 1 \leq i \leq N, \tag{3.34}$$

$$\psi_1(i) = 0. \tag{3.35}$$

2. Recursividade:

$$\delta_t(j) = \underset{1 \leq i \leq N}{max}[\delta_{t-1}(i)a_{ij}]b_j(O_t), \tag{3.36}$$

$$\psi_t(j) = \underset{1 \leq i \leq N}{argmax}[\delta_{t-1}(i)a_{ij}], \quad 2 \leq t \leq T, \quad 1 \leq j \leq N. \tag{3.37}$$

3. Término:

$$P^* = \underset{1 \leq i \leq N}{max}[\delta_T(i)], \tag{3.38}$$

$$q_T^* = \underset{1 \leq i \leq N}{argmax}[\delta_T(i)]. \tag{3.39}$$

4. Sequência de estados ótima

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \quad , t = T-1, T-2, \ldots, 1. \tag{3.40}$$

O algoritmo descrito tem, portanto, a propriedade de determinar a sequência de estados,

que maximiza a probabilidade $P(O^l|\lambda_l)$, para a $l-$ésima palavra [5, 87]. Assim, esse algoritmo pode ser usado para o "ajuste" da etapa de reconhecimento (identificação) e determinação da sequência de estados ótima do modelo.

A solução desses três problemas permite a elaboração de um sistema de reconhecimento automático da fala, utilizando HMM.

3.5 Considerações Gerais

Os modelos ocultos de Markov têm sido amplamente utilizados em sistemas de reconhecimento de fala, sendo até o momento, a modelagem que apresenta as melhores taxas de reconhecimento.

Neste capítulo, foi explicado o funcionamento do HMM, tanto para geração dos padrões de referência (etapa de treinamento) quanto para a identificação dos padrões de teste (etapa de reconhecimento).

Contudo, para a construção de um sistema de reconhecimento de fala contínua eficiente utilizando HMM, vários aspectos da modelagem do HMM devem ser considerados, como por exemplo, tipo do HMM (discreto, contínuo ou semi-discreto), número de estados, unidade básica (fones, trifones, palavras), entre outros.

Neste trabalho, foram realizados testes visando encontrar a configuração mais adequada a ser implementada em um sistema embarcado de baixo custo, ou seja, aquela que ofereça a melhor relação entre a taxa de reconhecimento e o custo computacional. Os resultados dos testes serão apresentados no Capítulo 5.

Capítulo 4

Reconhecimento de Fala em Sistemas Embarcados

Um sistema embarcado, ou sistema embutido, é um sistema microprocessado cujos *hardware* e *software* que o compõem são dedicados à aplicação que este executa. Diferente de computadores de propósito geral, um sistema embarcado realiza um conjunto de tarefas predefinidas, geralmente com requisitos específicos. Uma vez que o sistema é dedicado a essas tarefas específicas, é possível otimizar os projetos de *hardware* e *software* do mesmo, reduzindo, assim, o tamanho, recursos computacionais e custo do produto. Exemplos típicos de sistemas embarcados são aparelhos celulares, televisores, aparelhos de som, modem ADSL, brinquedos, câmeras digitais, mp3 *players*, fornos de microondas e outros aparelhos domésticos. Até mesmo carros incluem vários deles, na forma do sistema de injeção eletrônica, freio ABS, *airbag*, computador de bordo e outros sistemas de controle automotivos [97].

No que se refere à implementação em *hardware* de um sistema embarcado, diferentes soluções podem ser adotadas. Para aplicações com produção em grande escala (mercado de consumo), o mais adequado é o uso de um SoC (*System-On-a-Chip* – sistema em uma única pastilha) [24]. A arquitetura de *hardware* de um SoC embarcado pode conter um ou mais processadores, microcontroladores, DSP, memórias, interfaces para periféricos e blocos dedicados. Tais blocos, também chamados de IP Cores (*Intellectual Property Cores*), consistem de blocos em *hardware* que executam tarefas específicas e são definidos de modo a permitir o seu reuso em diferentes sistemas [98]. Os diversos IP Cores de um determinado SoC são interligados por uma estrutura de comunicação que pode variar de um barramento a uma rede complexa (NoC – *network-on-chip*) [99].

Algoritmos com bom desempenho na área de Processamento Digital de Sinais de Voz, em termos de taxa de reconhecimento, demandam um certo poder computacional, o que já se tem disponível em máquinas tipo PC em tempo real. No entanto, em sistemas embarcados dependentes de baterias, como um telefone celular, o desafio não representa apenas o desenvolvimento de *software*, mas também de *hardware* que se adapte às características inerentes de tais dispositivos (baixo consumo de energia, baixo poder de processamento, etc.) [10, 11].

Algumas técnicas de programação permitem a geração de um código mais eficiente no que se refere à quantidade de operações a serem executadas pelo processador (p.ex.: substituição de uma multiplicação por deslocamento de bits), o que consequentemente proporciona menor consumo de potência e energia. Sendo assim, a partir da otimização de algoritmos, utilização de estruturas de *hardware* específicas e um *pipeline* de instruções eficiente, é possível aumentar a velocidade, reduzir os recursos necessários, sem diminuir a taxa de reconhecimento.

Nas próximas seções, serão apresentados o processo de desenvolvimento de um sistema embarcado e alguns trabalhos que implementam reconhecimento de fala nesse contexto.

4.1 Processo de Desenvolvimento do *Hardware* de um Sistema Embarcado

Na Figura 4.1, é apresentado o fluxo de desenvolvimento do *hardware* de um sistema embarcado. Nas seções seguintes, será apresentado o papel de cada etapa.

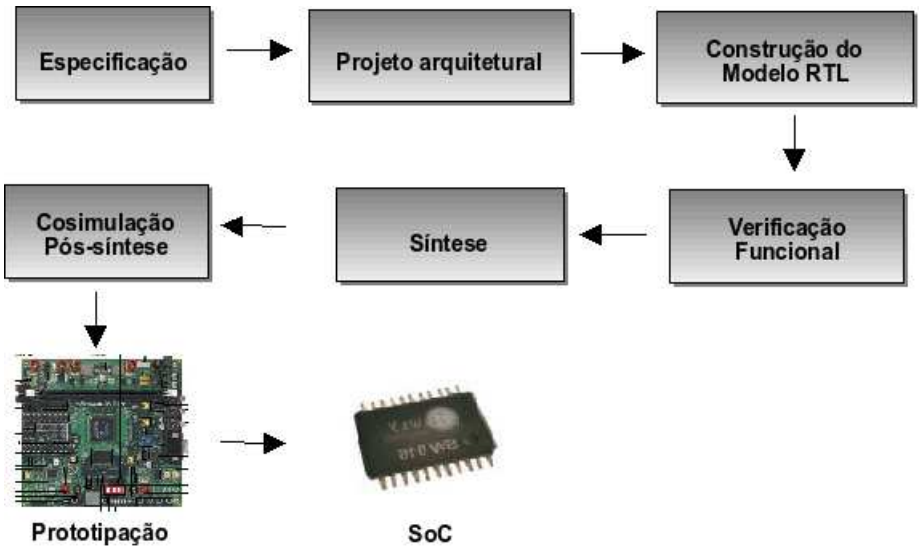


Figura 4.1: *Fluxo de desenvolvimento do hardware de um sistema embarcado.*

4.1.1 Especificação

O projeto de um sistema embarcado é iniciado usualmente por uma especificação da funcionalidade desejada, feita por meio de uma linguagem ou formalismo adequado. Idealmente, esta especificação deve ter um alto nível de abstração, no qual ainda não tenham sido tomadas decisões em relação à implementação dessa funcionalidade, em termos da arquitetura-alvo a ser adotada, nem sobre os componentes de *hardware* ou *software* a serem selecionados. Esta especificação deve ser preferencialmente executável, para fins de validação [24].

4.1.2 Projeto Arquitetural

A seguir, é feita uma exploração do espaço de projeto arquitetural, de modo a se encontrar uma arquitetura que implemente as funções contidas na especificação inicial e que atenda aos requisitos de projeto, em termos de custo, desempenho, consumo de potência, área, etc. O resultado final desta etapa é uma macro-arquitetura (ou arquitetura abstrata), contendo um ou mais processadores de determinados tipos (DSP, microcontroladores) e outros componentes necessários (memórias, interfaces, blocos dedicados de *hardware*), todos interconectados por meio de uma infra-estrutura de comunicação (um ou mais barramentos ou uma NoC).

Entre a especificação funcional e a macro-arquitetura, estabelece-se um mapeamento, a partir do qual cada função do sistema é atribuída a um processador ou a um bloco dedicado de *hardware*. Este mapeamento estabelece um determinado particionamento de funções entre *hardware* (blocos dedicados) e *software* (funções implementadas por um processador de instruções).

A exploração do espaço de projeto deve encontrar uma solução ótima para três questões básicas [24]:

1. Quantos e quais são os processadores e blocos dedicados de *hardware* necessários?
2. Qual é o mapeamento ideal entre funções e componentes de *hardware*?
3. Qual é a estrutura de comunicação ideal para conectar os componentes entre si, tendo em vista as trocas de informações que devem ser realizadas entre as funções mapeadas e os componentes?

Para que esta exploração seja efetuada rapidamente, é fundamental a existência de estimadores que, a partir da especificação funcional do sistema, sejam capazes de informar, com um grau de precisão adequado, os valores de métricas importantes de projeto (desempenho, consumo de potência, área) que irão resultar de cada alternativa arquitetural (uma macro-arquitetura e um mapeamento de funções).

Tendo em vista um espaço quase infindável de soluções arquiteturais possíveis, com uma correspondente complexidade computacional para exploração do mesmo, em busca de uma solução ótima ou mesmo subótima, a etapa do projeto arquitetural é usualmente simplificada pela escolha prévia de uma plataforma arquitetural conhecida e adequada ao domínio da aplicação. Essa arquitetura pode conter um ou mais processadores de tipos conhecidos, além de outros componentes necessários, todos interconectados por meio de uma estrutura de comunicação também pré-definida.

Usualmente, diversas funções serão mapeadas para um mesmo processador, sendo, então, implementadas como tarefas concorrentes que precisarão ser escalonadas e gerenciadas por um sistema operacional e, caso a aplicação assim o requisite, este deverá ser um sistema operacional de tempo real (RTOS). Além da função de escalonamento de tarefas, o RTOS deve oferecer recursos para comunicação entre as tarefas, considerando que estas poderão

estar distribuídas entre diversos processadores e mesmo blocos dedicados de *hardware*. Estes recursos devem oferecer uma abstração adequada ao *software* aplicativo, escondendo detalhes de mais baixo nível da infra-estrutura de comunicação. Também acionadores (*drivers*) dos periféricos devem ser oferecidos, ocultando, igualmente, detalhes das interfaces e da infra-estrutura de comunicação [24].

Uma vez definida a macro-arquitetura, é necessária a geração do *software* para a mesma, a partir da especificação funcional do sistema. Idealmente, seria desejável uma síntese automática do *software*, incluindo tanto o *software* aplicativo como o RTOS. Esta geração do *software* é bastante facilitada se a especificação funcional inicial tiver sido feita sobre uma interface de programação da aplicação padronizada (API – *Application Programming Interface*) que ofereça recursos para comunicação entre as tarefas e para a qual exista uma implementação sobre a plataforma arquitetural (processadores e RTOS) selecionada. É também necessário um compilador, que traduza a especificação funcional para uma linguagem de programação adequada a cada processador adotado (a menos que a especificação funcional já tenha sido feita em uma determinada linguagem).

Componentes de *hardware* e *software* selecionados para a macro-arquitetura podem ter interfaces heterogêneas, implementando diferentes protocolos de comunicação. Neste caso, é necessária a síntese da comunicação entre os componentes. Esta síntese deve gerar adaptadores (*wrappers*) que fazem a conversão entre os diferentes protocolos. Adaptadores de *software* podem ser considerados como elementos de um RTOS dedicado gerado para a aplicação, enquanto que adaptadores de *hardware* são componentes dedicados que ajustam as interfaces dos componentes ao protocolo da infra-estrutura de comunicação (embora conexões ponto-a-ponto também sejam possíveis).

4.1.3 Construção do Modelo RTL

O modelo RTL (*Register Transfer Level*) é a descrição da especificação do projeto em nível de fluxo de dados entre registradores, controlado por um clock. Esse modelo geralmente é escrito em uma linguagem de descrição de *hardware*, como VHDL ou Verilog.

4.1.4 Verificação Funcional

A melhoria contínua das metodologias de projeto e processos tem possibilitado a criação de grandes e complexos sistemas digitais. A verificação funcional é uma das principais tarefas do fluxo de desenvolvimento, e tem o objetivo de verificar todas as funcionalidades de projeto e assegurar que estas estão ocorrendo da maneira especificada. No contexto de *Intellectual Property* (IP) *cores design*, a verificação funcional é um elemento chave para garantia de sucesso em seu reuso [100, 101].

Esta tarefa chega a consumir cerca de 70% do tempo total de desenvolvimento [100, 102, 103, 104]. Diante disso, ferramentas que possibilitem a geração rápida e eficiente de um ambiente de simulação podem e devem ser utilizadas.

Ao longo do processo de projeto, inúmeras descrições do sistema embarcado são geradas, em diferentes níveis de abstração (especificação funcional, macro-arquitetura, micro-arquitetura), cobrindo diferentes aspectos (*software* e *hardware*) e eventualmente com a utilização combinada de diferentes linguagens (e.g. Java e VHDL). Estas descrições precisam ser validadas, o que exige a execução das descrições, no caso de linguagens de programação como Java e C, ou sua simulação, no caso de linguagens de descrição de *hardware* ou de sistemas como VHDL e SystemC [24].

Nenhuma dessas linguagens consegue cobrir simultaneamente todos os domínios de aplicação, níveis de abstração e aspectos de *hardware* e *software*. Assim, é bastante comum a necessidade de validação de descrições multi-linguagem em determinados passos do projeto. Um exemplo evidente é a validação combinada da microarquitetura descrita, por exemplo, em VHDL, e do *software* que irá rodar sobre um processador. Outro exemplo é a validação de um sistema embarcado contendo partes de *hardware* digital, descritas em VHDL, *hardware* analógico, descrito por equações diferenciais em Matlab, e *software*, descrito em C.

Por outro lado, a validação de um sistema muito complexo pode ser bastante facilitada por um processo de refinamentos sucessivos, no qual apenas partes selecionadas do sistema são descritas num nível de abstração mais detalhado (e.g. micro-arquitetura), enquanto o restante do sistema permanece descrito de forma mais abstrata (p.ex. como macro-arquitetura ou mesmo como componente puramente funcional). Esta abordagem, que permite focalizar melhor as decisões de projeto que precisam ser validadas a cada novo passo de refinamento, também pode resultar em descrições multi-linguagem.

A simulação de sistemas que possuem módulos descritos em diferentes linguagens é chamada de cosimulação. Exemplos de cosimulação entre VHDL e C são encontrados nos simuladores VCI [105] e SIMOO [106]. Ferramentas de cosimulação atuais, como CoCentric System Studio, da Synopsys [107], e Seamless CVE, da Mentor [108], permitem a integração de SystemC, C, simuladores de *software* no nível de instruções de máquina de um processador e HDL diversas, como VHDL e Verilog. MCI [109] é um exemplo de solução genérica, que permite a geração de modelos de cosimulação a partir de uma especificação multi-linguagem do sistema. No entanto, a cosimulação dá-se a partir de um mecanismo de comunicação proprietário, como nas soluções comerciais. O ambiente Ptolemy [110] adota uma abordagem orientada a objetos na modelagem do sistema e oferece um conjunto de classes explicitamente orientado à cosimulação de diferentes modelos de computação.

No contexto de reuso de componentes IP, torna-se interessante o desenvolvimento de simulações distribuídas, nas quais componentes IP sejam simulados remotamente, no site de seus fornecedores, visando a proteção da propriedade intelectual. O ambiente JavaCAD [111] oferece este recurso, mas está restrito a componentes descritos em Java, assim como o ambiente proposto em [112] está restrito a componentes VHDL e Verilog. Soluções mais genéricas, abertas a diversas linguagens, são propostas pelos ambientes WESE [113], baseado em CORBA, e DCB [114], inspirado no padrão HLA [115] de simulação distribuída.

A validação por simulação apresenta duas grandes restrições. Em primeiro lugar, o nú-

mero de casos de testes para uma validação exaustiva da descrição do sistema é muito grande, obrigando os projetistas na prática a limitarem-se a uma cobertura parcial dos mesmos. Em segundo lugar, quanto mais baixo o nível de abstração, maior o número de casos de teste e mais demorada é a simulação, pelo maior detalhamento da descrição. Técnicas de verificação formal [116], que realizam uma validação simbólica do sistema, e não numérica, prometem resolver este problema por cobrirem todos os possíveis casos de teste a partir de um único processamento. No entanto, tais técnicas ainda não atingiram um grau de maturidade suficiente que permita a sua aplicação em grande escala em todo o processo de projeto, estando limitadas a determinadas combinações de linguagens, estilos de descrição, domínios de aplicação e níveis de abstração.

4.1.5 Síntese

Uma vez definidos e validados os componentes de *hardware* da macro-arquitetura, incluindo a infra-estrutura de comunicação e os eventuais adaptadores, poderá ser feita a síntese do *hardware*. Em uma primeira etapa, a macro-arquitetura pode ser expandida para uma micro-arquitetura (ou arquitetura RTL), contendo o detalhamento de todos os componentes e suas interconexões, pino-a-pino e considerando o funcionamento do circuito com precisão de ciclo de clock. Em uma segunda etapa, podem ser usadas ferramentas convencionais para síntese de *hardware*, que a partir da micro-arquitetura irão gerar o layout final do circuito. Para tanto, é necessário que a microarquitetura esteja descrita em uma linguagem apropriada para estas ferramentas, como VHDL ou Verilog. A existência prévia de layouts para os componentes de *hardware* selecionados facilita bastante esta síntese, que se limita então ao posicionamento e roteamento de células.

4.1.6 Cosimulação Pós-síntese

Na etapa de síntese, é definido qual o dispositivo que será utilizado para prototipação do sistema, o que implica que, a partir de então, aspectos específicos do dispositivo, como por exemplo, atrasos das portas lógicas passam a ser considerados durante a execução. Isso torna necessária uma nova cosimulação pós-síntese de todo o sistema, de modo a assegurar a manutenção de seu correto funcionamento.

4.1.7 Prototipação

Vencida a etapa de cosimulação pós-síntese, pode-se passar para etapa final, a prototipação. Essa etapa consiste na implantação do código gerado pela síntese em algum dispositivo de *hardware*.

Após os testes em *hardware* e seu correto funcionamento de acordo com a especificação, pode ser realizada a fabricação do CI, que consistirá, por exemplo, em um SoC.

4.2 Trabalhos Relacionados

Muitos trabalhos vêm sendo desenvolvidos no contexto de processamento digital de sinais de voz para sistemas embarcados. Nesta seção, será realizada uma breve descrição de alguns deles.

Para apresentação dos trabalhos, será utilizada uma sequência cronológica, no intuito de mostrar um pouco da evolução das pesquisas desenvolvidas na área de reconhecimento de fala em sistemas embarcados.

Em [117] (Kim 1996) foi desenvolvido um *chip* para reconhecimento de palavras isoladas, realizando detecção de início e fim de palavras, extração de vetores de características e cálculo da distância espectral por meio da técnica de Alinhamento Dinâmico no Tempo (DTW - *Dynamic Time Warping*). O chip foi desenvolvido utilizando a tecnologia CMOS 0,8 μ m com 66.760 portas lógicas e *clock* de 10MHz. Podem ser reconhecidas 1000 palavras isoladas por segundo com uma taxa de acerto de 90,3%.

Em [118] (Yuanyuan 2001) foi utilizado o microcontrolador 8051 para desenvolver um sistema de reconhecimento de até 20 frases de aproximadamente 1 segundo. A arquitetura de *hardware* do SoC é composta de: um núcleo MCU de 8-bits, memória RAM (*Random Access Memory*) interna de 512 bytes, memória ROM (*Read Only Memory*) interna de 8 kbytes, 72192x6 bits de memória ROM para voz, conversor AD/DA (Analógico-Digital/Digital-Analógico) de 10 bits, módulo PWM (*Pulse Width Modulation*), portas de I/O e alguns outros circuitos periféricos. O sistema completo foi testado em FPGA com algumas frases e pequenas sentenças, com três locutores diferentes. A taxa de amostragem utilizada foi 8 KHz e a taxa de reconhecimento obtida é de aproximadamente 96% em um ambiente cuja relação sinal-ruído seja superior a 10dB e de 90% abaixo disso.

[119] (Nakamura 2001) consiste de um chip de reconhecimento para monossílabos que utiliza HMM como técnica de classificação. O sistema executa a análise LPC do sinal de voz, a quantização vetorial e por fim a classificação por meio do método HMM. O tamanho dos quadros de voz é de 11,6 ms, o qual inclui 256 amostras de 16 bits. A frequência de amostragem é 22,05 KHz e a dimensão do vetor de características extraído é 24 com elementos de 32 bits. Os parâmetros de HMM são calculados via *software* e armazenados na memória externa. A linguagem de descrição de *hardware* utilizada foi VHDL. Na etapa de síntese, foi utilizada a ferramenta Synopsys Design Compiler [Synopsys 2009] e o *layout* do *chip* foi gerado com a ferramenta Avant! Apollo.

[120] (Vargas 2001) apresenta uma abordagem para implementação do algoritmo de Viterbi, utilizado para calcular a verossimilhança (*likelihood score*) no HMM. Consiste de uma implementação *hardware/software* co-design, na qual o algoritmo de Viterbi é implementado em *hardware* juntamente com a estrutura do HMM. São construídas máquinas de estados probabilísticas que executam em processos paralelos, sendo um processo para cada palavra do vocabulário do sistema. Com isso, é obtido um aumento de aproximadamente 500 vezes na velocidade de execução, quando comparada à implementação clássica de Viterbi. Uma

das idéias principais do trabalho é construir a arquitetura do HMM utilizando apenas somadores, comparadores e operadores lógicos e realizar a correspondência dos padrões (*Pattern Matching*) sem a utilização do algoritmo de Viterbi. Na abordagem apresentada, o processo de reconhecimento é realizado pela pontuação das probabilidades das sequências obtidas na quantização vetorial (semelhante ao algoritmo Viterbi), sendo que essa pontuação pode ser realizada por operações executadas diretamente pelos blocos lógicos contidos no próprio FPGA. A taxa de reconhecimento obtida foi de 99,7% em um vocabulário de 10 palavras.

[26] (Kim 2002) consiste de um sistema de reconhecimento de palavras isoladas em ambientes com ruído. Para extração de características é utilizado um modelo modificado da técnica *Zero-Crossings with Peak Amplitude* (ZCPA) e para classificação o modelo *Radial Basis Function* (RBF) de redes neurais. A justificativa para utilização de redes neurais como técnica de classificação é que esta oferece uma boa taxa de reconhecimento e rejeição de palavras não conhecidas pelo vocabulário, além da sua arquitetura regular que permite uma implementação mais simples em *hardware*. Nos testes realizados, essa técnica se mostra eficiente para o reconhecimento de palavras isoladas em um pequeno vocabulário, no entanto, isso não acontece para padrões dinâmicos. As taxas de reconhecimento são de 95,2% a 95,6%.

[121] (Vargas 2002) é baseado em [120] e apresenta uma abordagem para aumentar a taxa de reconhecimento em um ambiente com ruído. A idéia principal do trabalho consiste em realizar a construção do sinal de entrada, todas as vezes que a sequência de code labels gerada não respeitar uma regra fundamental de qualidade. O algoritmo utilizado é baseado em HMM e realiza um teste *on-line* por meio de uma versão modificada da técnica de reconstrução de blocos (*Recovery Block*). Nos testes realizados foram utilizadas 16 palavras pronunciadas por 100 pessoas diferentes obtendo-se taxas de reconhecimento de até 66%. Também foram utilizadas três arquiteturas: a primeira em um microcomputador, a segunda no Texas DSP TMS-320C67 e a terceira, uma arquitetura baseada em *hardware-software codesign* composta pelo mesmo DSP da segunda implementação juntamente com o FPGA FLEX10K20 da Altera.

[122] (Marcus 2005) apresenta um coprocessador baseado no sistema de reconhecimento Sphinx [123] da Carnegie Mellon University (CMU) [124], e tem como maior objetivo desenvolver uma implementação simples do algoritmo de *Baum-Welch*, possibilitando assim sua fácil implantação em um FPGA. O algoritmo de *Baum-Welch* é utilizado para cálculo de probabilidades gaussianas, tanto na etapa de treinamento, quanto de reconhecimento, contudo, neste trabalho, é dada maior ênfase à primeira. A justificativa para otimização deste algoritmo se deve ao consumo de tempo demandado pela etapa de estimação de gaussianas, que no sistema Sphinx chega a 49,8% do total, podendo ser de até 70% em outros sistemas [125]. Apesar da tabela de Gaussianas e o modelo serem iguais nas etapas de treinamento e reconhecimento, a largura de banda utilizada na primeira é maior, uma vez que, enquanto na segunda é realizada apenas uma busca na tabela para gerar uma pontuação, na primeira os valores são propagados para frente e para trás para que possam ser gerados. São calculadas as gaussianas de um conjunto de vetores de mesmo tamanho, 39 elementos no caso do

SPHINX. No entanto, esse tamanho pode ser ajustado de 3 a 63 elementos, com base em um registrador específico. O desempenho do sistema desenvolvido foi comparado ao desempenho de um PC, e obteve um aumento de 1.76 na velocidade de execução.

[126] (Schuster 2006) apresenta o projeto em FPGA de um coprocessador para modelagem acústica em tempo real, baseada no sistema de reconhecimento de fala Sphinx 3. Esse trabalho considera que um sistema de reconhecimento de fala é composto pelos quatro seguintes módulos: Extração de Características (EC), Modelagem Acústica (MA), Classificador de Fonemas (CF - *Phoneme Evaluator*) e Modelagem de Palavras (MP). O módulo MA tem a tarefa de avaliar as saídas de EC com base em uma base de dados de probabilidade gaussianas e gerar um conjunto de *scores* (senones). O módulo CF, por sua vez, associa grupos de senones em HMM representando as unidades fonéticas (fonemas), disponíveis no dicionário do sistema. Por fim, o módulo MP utiliza uma estrutura baseada em árvores para ligar os fonemas, formando palavras, com base nas sequências definidas no dicionário. Os valores necessários para o cálculo dos senones são armazenados em memórias ROM externas, que podem ser trocadas ou reescritas, no caso, em que se deseje utilizar um outro *corpus* de fala.

[27] (Lim 2006) Implementa um sistema de reconhecimento no *softcore* MicroBlaze da Xilinx [127]. *Softcore* MicroBlaze é um IP Core escrito em uma HDL, desta forma, ele é configurado escolhendo-se quais os componentes que serão incluídos no sistema, de acordo com o tipo de aplicação a ser desenvolvida, diminuindo o consumo de energia e a área utilizada [127]. O processo de reconhecimento implementado inclui três etapas: extração de características, cálculo da probabilidade de emissão e a busca utilizando o algoritmo de Viterbi. Na etapa de extração de características, para cada quadro, são calculados 13 MFCC juntamente com suas primeira e segunda derivadas. O cálculo da probabilidade de emissão emprega o modelo de fonema que só é afetado pelos fonemas vizinhos dentro dos limites da palavra. Cada modelo de fonema consiste de três estados e cada estado possui densidades probabilidade de transição e de saída, modeladas por mistura gaussiana. Os modelos foram treinados utilizando o corpus TIMIT e aproximadamente 4000 misturas gaussianas.

[28] (Lin 2007) refere-se ao projeto Silico Vox apresentado em [128]. O projeto tem o objetivo de construir um sistema completo de reconhecimento de fala em silício, utilizando como modelo de referência o sistema Sphinx 3 [123], apresentado no Capítulo 5. As principais estratégias adotadas para permitir a implementação do sistema em silício sem comprometer a eficiência do processo de reconhecimento foram:

- *largura de bits:* os valores em ponto-flutuante de 32 bits utilizados entre as etapas de *frontend* e GMM foram substituídos por valores com menos bits. Com essa redução na largura dos bits (33%), é obtida uma redução na área do *chip*, como também, a quantidade de memória necessária (50%).

- *Log lookup table*: para calcular a probabilidade logarítmica de cada nó, é calculada a probabilidade logarítmica de cada mistura gaussiana e em seguida, realizada a soma

com aproximadamente 100.000 elementos de uma *lookup table*. Uma vez que, armazenar esses elementos em um *chip* resulta em custo elevado de *hardware*, é utilizada uma interpolação com quatro polinômios de terceira ordem.

- *Pruning threshold*: no Sphinx 3, inicialmente, são atualizadas as probabilidades de todos os HMM ativos, com base na probabilidade mais alta são determinados os limiares de corte de cada quadro, e por fim, é realizada a podagem dos HMM ativos. Esse ciclo, requer duas vezes a passagem pelos HMM ativos, o que praticamente dobra o acesso à memória e compromete o desempenho do sistema. Neste algoritmo, as transições também só são calculadas durante a segunda passagem pelos HMM. Para reduzir a quantidade de acessos à memória, o algoritmo foi modificado de forma que é realizada apenas uma passagem pelos HMM, e o limiar de corte é definido com base na melhor pontuação do quadro anterior. Isso também permite a utilização de paralelismo, no qual diferentes HMM se encontram em etapas diferentes do processo simultaneamente.

[129] (Ke 2008) apresenta o desenvolvimento de um sistema de reconhecimento de fala baseado em HMM, propondo o uma implementação para o algoritmo *forward* em FPGA. Foram utilizados 16 coeficientes LPCC na etapa de extração de características, e o algoritmo LBG para geração do *codebook*, contendo 128 vetores de código. O processador utilizado é o Microblaze [127]; OPB é o barramento do chip que é utilizado para acessar os recursos do sistema fornecendo um barramento de 32 bits para endereçamento e 32 bits para dados; o controlador SPI é responsável pela comunicação com o modelo de captura da fala; OPB-INC realiza a função de um serviço de interrupção SPI como o GPIO; a Flash é utilizada para armazenar o programa de inicialização do sistema e os modelos gerados depois do treinamento.

Em [29] (Miura 2008) é apresentado um processador GMM para reconhecimento de fala contínua com grande vocabulário em tempo real. No algoritmo do HMM, cada HMM corresponde a um fone, cada palavra é expressa como uma sequência de fones e cada sentença é representada por uma sequência de palavras. No algoritmo de Viterbi são utilizados logaritmos com o objetivo de evitar *underflow*. Apenas o módulo de GMM foi avaliado em FPGA, os demais foram avaliados somente em software. Para o módulo GMM foi obtida uma redução de 89,8% na frequência de operação (para 30.4 MHz) e de 84,2% na largura de banda para memória (47 Mbps).

Diferentemente dos trabalhos anteriores, que desenvolveram uma arquitetura dedicada para reconhecimento de fala em dispositivos com limitações no que se refere à capacidade de processamento e memória, [130] (Acero 2008) apresenta uma interface baseada em reconhecimento de fala em um celular, na qual a única tarefa do celular é captar o sinal e enviar para um servidor que realiza todo o processo de reconhecimento, retornando seu resultado. O sistema consiste de uma aplicação chamada Live Search for Mobile (LS4M) desenvolvida para celulares que possuem o sistema operacional Windows Mobile. A aplicação permite o acesso a diferentes portais de informação baseados na Web, como localização de hotéis,

restaurantes, rotas, e outros.

Na Tabela 4.1 são apresentadas as principais características dos trabalhos apresentados neste capítulo, bem como alguns outros pesquisados.

Tabela 4.1: *Principais características dos trabalhos relacionados.*

	Trabalho	Tamanho do Vocabulário	Taxa de reconhecimento	Modelagem Acústica	Modelagem Linguistica	Classificação	Arquitetura	Principal otimização
1.	[Veitch 2011]	Fala contínua	80%	MFCC	N-gram	WFST	FPGA	GMM
2.	[Cheng 2011]	Fala contínua	93%	MFCC	N-gram	WFST	HW-SW Co-design	GMM
3.	[Choi 2010]	Fala contínua	88%	MFCC	N-gram	HMM	FPGA	GMM + Viterbi
4.	[Lin 2009]	Fala contínua	93%	MFCC	N-gram	HMM	FPGA	Viterbi
5.	[Zhou 2009]	Dígitos isolados	94%	MFCC	-	HMM	FPGA	Viterbi
6.	[Amudha 2008]	Dígitos isolados	100%	MFCC + SOFM [Huang 1992]	-	Multi Layer Perceptron	FPGA	Algoritmo Back Propagation
7.	[Miura 2008]	Fala contínua (20.000 palavras)	92,6%	GMM	N-gram	HMM	FPGA	GMM
8.	[Acero 2008]	Fala contínua	60% a 65%	Não informa	Não informa	Não informa	Distribuído	-
9.	[Ke 2008]	Não informa	Não informa	LPCC	-	HMM	Microblaze + FPGA	Algoritmo forward
10.	[Lin 2007]	Fala continua (1.000 palavras)	93,3%	MFCC + GMM	N-gram	HMM	FPGA	GMM e Viterbi

Continua na próxima página...

Tabela 4.1 – Continuação

	Trabalho	**Tamanho do Vocabulário**	**Taxa de reconhecimento**	**Modelagem Acústica**	**Modelagem Linguistica**	**Classificação**	**Arquitetura**	**Principal otimização**
11.	[Lin 2006]	Fala contínua	Não informa	MFCC + Delta Cepstrais + GMM	-	HMM	Microblaze + FPGA	Viterbi
12.	[Schuster 2006]	Fala contínua	Não informa	LPCC	-	HMM	FPGA	
13.	[Yoshizawa 2006]	Palavras isoladas (800 palavras)	Entre 98% e 99,7%	MFCC + GMM	-	HMM	FPGA	Tratamento de ruído e HMM
14.	[Marcus 2005]	Não informa	Não informa	Não informa	Não informa	HMM	FPGA	Algoritmo Baum-Welch
15.	[Nedevschi 2005]	Palavras isoladas	97%	LPCC + GMM	-	HMM	FPGA	Diferentes idiomas e Low Power
16.	[Phadke 2004]	Palavras isoladas	Até 99,9%	MFCC	-	DTW	DSP	Detecção de início e fim de palavras
17.	[Vaidhyanathan 2004]	4 monossílabos (A, B, C,...)	75%	LPCC	-	HMM	Simulação	

Continua na próxima página...

Tabela 4.1 – Continuação

	Trabalho	**Tamanho do Vocabulário**	**Taxa de reconhecimento**	**Modelagem Acústica**	**Modelagem Linguistica**	**Classificação**	**Arquitetura**	**Principal otimização**
18.	[Mathew 2003]	Fala contínua	Não informa	LPCC	n-gram	HMM	FPGA	GMM
19.	[Melnikoff 2002]	Fala contínua	50%	Não informa	-	HMM	PC + FPGA	Viterbi
20.	[Vargas 2002]	16 palavras isoladas	Até 66%	Não informa	-	HMM	DSP e FPGA	Tratamento de ruído
21.	[Kim 2002]	Palavras isoladas	95,2% a 95,6%	Zero-Crossings with Peak Amplitude (ZCPA)	-	Modelo Radial Basis Function (RBF) de redes neurais	FPGA	ZCPA
22.	[Cipriano 2001]	Palavras isoladas	Não informa	MFCC	-	HMM	FPGA	
23.	[Vargas 2001]	10 palavras isoladas	99,7%	LPC + Coeficientes Cepstrais	-	HMM	Microprocessador + FPGA	Viterbi
24.	[Nakamura 2001]	Monossílabos	Não informa	LPC	-	HMM	FPGA	

Continua na próxima página...

Tabela 4.1 – Continuação

	Trabalho	**Tamanho do Vocabulário**	**Taxa de reconhecimento**	**Modelagem Acústica**	**Modelagem Linguistica**	**Classificação**	**Arquitetura**	**Principal otimização**
25.	[Yuanyuan 2001]	Frases de até 1 segundo	96,4%	LPCC	-	DTW (Dynamic Time Warping)	Microcontrolador 8051	
26.	[Gong 2000]	Dígitos contínuos	Não informa	MFCC	-	HMM	DSP	
27.	[Kim 1996]	Palavras isoladas	90,3%	LPC	-	DTW (Dynamic Time Warping)		

4.3 Considerações Gerais

Com base no que foi apresentado nos capítulos anteriores, é possível verificar a partir dos trabalhos mais atuais uma tendência para o reconhecimento de fala contínua, diferente dos mais antigos que tratavam apenas de palavras isoladas. Diante desse contexto, nota-se a inclusão de modelos linguísticos nos sistemas de reconhecimento, uma vez que, apenas com modelos acústicos, essa tarefa se torna complexa e pouco eficiente. O modelo linguístico predominante é o *n-gram*.

No que se refere aos parâmetros acústicos, os mais utilizados são os LPCC e MFCC. Outra característica que se pode observar na maioria dos trabalhos é a utilização de GMM durante a etapa de clusterização, que antecede a classificação. Por ser esta uma etapa que requer muito processamento, torna-se alvo de muitas propostas de otimização.

Na etapa de classificação, o uso de HMM é predominante, e os poucos trabalhos que utilizam Redes Neurais ou DTW, realizam reconhecimento de palavras isoladas com pequeno vocabulário. Para esta etapa, muitos trabalhos propõem utilização de paralelismo e memória cache, principalmente no algoritmo de Viterbi.

Vale a pena salientar que entre os trabalhos pesquisados, apenas um trata do idioma português, o que ressalta mais uma contribuição desta tese de doutorado.

Um recurso útil utilizado em alguns trabalhos é o de *softcores*, que facilitam a descrição em HDL dos elementos que compõem o sistema.

Outro fator em comum entre vários trabalhos é a utilização do sistema Sphinx como modelo de referência.

Nos próximos capítulos, será apresentado o processo de avaliação dos parâmetros e configurações empregados em um sistema de reconhecimento de fala contínua utilizando MFCC e HMM. Foram realizados testes no *framework* Sphinx, com objetivo de encontrar a configuração que apresente a melhor relação taxa de reconhecimento *versus* custo computacional, de modo a permitir sua implementação em um sistema embarcado com recursos computacionais limitados.

Capítulo 5

Modelagem do Ambiente de Desenvolvimento

Um sistema de reconhecimento de fala contínua com HMM contínuo consiste, basicamente, das etapas apresentadas na Figura 5.1 e descritas a seguir.

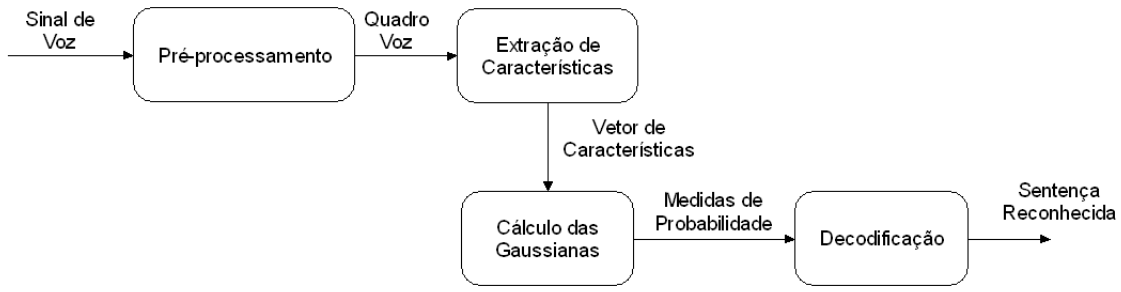


Figura 5.1: *Etapas de um sistema para reconhecimento de fala contínua com HMM contínuo.*

1. Pré-processamento - responsável por preparar e extrair do sinal de voz os dados relevantes e menosprezar a informação redundante, assim como dividir o sinal de voz em quadros que atendam o limite de estacionariedade;

2. Extração de características - para cada quadro do sinal de voz, é extraído um vetor de características, que será comparado com os padrões de referência, conhecidos pelo sistema;

3. Cálculo das gaussianas - nesta etapa é calculada a probabilidade que o vetor de características obtido tenha sido gerado por cada um dos padrões de referência conhecidos pelo sistema;

4. Decodificação - são utilizadas as probabilidades obtidas na etapa anterior, juntamente com o Modelo Linguístico (ML) para decidir se o padrão de teste é conhecido pelo sistema e caso seja, a qual padrão de referência corresponde.

Para permitir a implementação dessas etapas em um sistema embarcado de baixo custo, torna-se necessária a otimização e adaptação das técnicas empregadas, de forma a proporcionar uma redução dos custos de *hardware*. Para atingir esse objetivo, neste trabalho foram realizados experimentos e análises, de modo a identificar possíveis adaptações, a partir de simplificações matemáticas e redução de parâmetros nas etapas citadas.

A metodologia empregada para desenvolvimento das análises e experimentos realizados é descrita na Seção 5.1 deste capítulo. Na Seção 5.2, é apresentado o *framework* utilizado para realização dos experimentos. Na Seção 5.3, é apresentada a base de dados utilizada, e em seguida, na Seção 5.4, as adaptações que foram necessárias para permitir o uso dessa base no *framework* citado. Na Seção 5.5, são descritos os cenários de testes, com seus respectivos resultados na Seção 5.6. Nas Seções 5.7 e 5.8, são descritas as análises e simplificações dos algoritmos e equações envolvidas, nas etapas de Cálculo das Gaussianas e Decodificação, respectivamente.

5.1 Metodologia Empregada

Como foi discutido anteriormente, para permitir a implementação das etapas de um sistema de reconhecimento para fala contínua, em um sistema embarcado de baixo custo, torna-se necessária a adaptação das técnicas empregadas, de forma a proporcionar uma redução dos custos de *hardware*. Com o objetivo de identificar possíveis adaptações e simplificações, neste trabalho, foi empregada a metodologia descrita a seguir.

1. Definição das características acústicas a serem utilizadas

   Dois métodos de análise espectral predominam nos sistemas de reconhecimento de fala: o método de análise espectral LPC (*Linear Predictive Coding*) [1, 2, 41, 42, 43, 44] e o método de análise espectral por banco de filtros [45, 46, 47]. Nesta etapa, foi realizado um estudo do estado da arte das técnicas citadas para o fim de reconhecimento de fala contínua. Ao final desse estudo, os coeficientes mel-cepstrais foram apontados como os que oferecem melhor taxa de reconhecimento [30, 31, 32, 33, 34, 36, 45, 46, 47, 131, 132, 133, 134]. Diante disso, foram realizados testes variando o número de filtros e coeficientes, de forma a encontrar a melhor configuração a ser utilizada.

2. Definição da configuração do HMM

   Um dos objetivos deste trabalho consiste na utilização de HMM para construção do modelo acústico e classificação. Contudo, vários parâmetros podem ser variados nesse modelo, dentre os quais: tipo do HMM (discreto, contínuo ou semi-contínuo) e número de estados. Nesta etapa, também foi realizado um estudo do estado da arte, ao final do qual foi verificado que os melhores resultados para fala contínua são obtidos com HMM contínuos [30, 31, 32, 33, 34, 36]. Diante disso, foram efetuados testes para identificar o melhor número de estados a ser utilizado para o português brasileiro.

3. Definição da técnica para construção do modelo linguístico

   Modelos linguísticos (ML) capturam regularidades na linguagem falada e são utilizados no reconhecimento para estimar a probabilidade da sequência de palavras. Em um sistema de reconhecimento de palavras isoladas, é possível obter boas taxas de reconhecimento sem o uso de um ML, contudo este torna-se fundamental em sistemas de reconhecimento de fala contínua.
   Existem várias técnicas para construção desses modelos, dentre as quais: N-gram, árvores de decisão e gramática livre de contexto.
   Mais uma vez, foi realizado um levantamento na literatura da relação entre cada técnica e sua eficiência no processo de reconhecimento. Um dos modelos linguísticos que tem fornecido melhores resultados é o N-gram [9, 30, 32, 33, 34, 36, 38, 39, 62, 63], o qual foi escolhido para ser utilizado neste trabalho.

4. Avaliação do impacto dos parâmetros empregados em relação à taxa de reconhecimento *versus* tempo de execução

   Como foi discutido anteriormente, diversos parâmetros podem ser variados na configuração de um sistema de reconhecimento de fala utilizando coeficientes mel-cepstrais, HMM e modelagem linguística N-gram. A variação desses parâmetros pode aumentar ou diminuir a taxa de reconhecimento em nome de uma maior ou menor carga computacional. Neste trabalho, foi investigado o impacto dos seguintes parâmetros na taxa de reconhecimento:

   - Número de estados do HMM;
   - Número de filtros para cálculo dos coeficientes mel-cepstrais;
   - Número de coeficientes mel-cepstrais;

   Para se obter uma idéia do impacto em relação à carga computacional de cada configuração utilizada, foi medido o tempo de execução das seguintes etapas do sistema:

   - Extração das características para o treinamento;
   - Treinamento;
   - Extração das características para o reconhecimento;
   - Reconhecimento.

   O resultados obtidos serão apresentados neste capítulo, juntamente com as decisões tomadas a partir desses resultados.

5. Análise das etapas de Cálculo das Gaussianas e Decodificação

   Para as etapas de Cálculo das Gaussianas e Decodificação, foram realizadas análises

das equações matemáticas e dos algoritmos empregados nas mesmas, de modo a encontrar simplificações e adaptações para o contexto embarcado. Essas simplificações e adaptações permitem a construção de um *hardware* mais simples e, consequentemente, mais barato.

Com base nos resultados obtidos e apresentados no restante deste capítulo, foi possível definir:

- Número de coeficientes (MFCC) utilizado;
- Número de filtros do banco de filtros triangulares;
- Número de estados do HMM;
- Simplificações matemáticas para o cálculo das gaussianas;
- Adaptação do algoritmo utilizado na etapa de Decodificação.

5.2 Suporte ferramental

Atualmente, estão disponíveis alguns sistemas abertos, direcionados à pesquisa em reconhecimento de fala que implementam o estado da arte na área. Estes sistemas são frequentemente utilizados nas avaliações do DARPA (*Defense Advanced Research Projects Agency*), além de serem o ponto de partida para a maioria dos pesquisadores da área e de áreas afins. A seguir, são descritos rapidamente os principais sistemas referenciados.

O HTK (*Hidden Markov Model Toolkit*) [135] é definido como um conjunto de ferramentas de *software* para construção e manipulação de HMM. Teve sua primeira versão desenvolvida em 1989 pelo professor Steve Young do Departamento de Engenharia da Universidade de Cambridge (CUED - *Cambridge University Engineering Department*). É usado principalmente para pesquisa em reconhecimento de voz, porém é também bastante utilizado em outras aplicações, como pesquisas em síntese de voz, reconhecimento de caracteres e sequenciamento de DNA. O HTK é uma das ferramentas mais utilizadas e divulgadas da área. Consiste em conjunto de bibliotecas e ferramentas com código-fonte disponíveis em C, porém compatíveis com C++. As ferramentas contidas nele provêem funcionalidades para análise da fala, treino de HMMs, testes e análise de resultados. O software suporta tanto misturas de Gaussianas com densidade contínua, bem como distribuições discretas e pode ser utilizado para construir sistemas complexos baseados em HMMs. Contém documentação e exemplos em um documento conhecido como "HTK Book", bem como um fórum de discussão disponível no sítio da ferramenta. Sua licença é restrita apenas ao uso não comercial.

O ATK (API for HTK) [136] é uma interface de programação para o HTK e foi criado para facilitar o desenvolvimento de aplicações experimentais baseadas no mesmo. Consiste em uma camada de programação orientada a objeto, em C++, que faz acesso às bibliotecas padrão do HTK. Isso permite que novos reconhecedores construídos a partir de versões

customizadas do HTK sejam compilados com o ATK e, então, testados em sistemas já em funcionamento. Assim como o HTK, o ATK é compatível com as plataformas de sistemas operacionais Unix® e Microsoft Windows® (a partir de linha de comando e, apenas no Windows, disponibiliza interface gráfica). O ATK roda com múltiplos processos em paralelo (multi-threaded) e permite que múltiplas entradas de comandos (por voz, clique do mouse, gestos, etc.) sejam combinadas em um mesmo pacote de dados. O ATK pode retornar os resultados de reconhecimento de voz palavra-a-palavra, assim que são reconhecidas, de forma a reduzir o atraso a um mínimo, permitindo que sistemas de resposta rápida sejam construídos.

O principal objetivo do pacote de ferramentas do ISIP (*Institute for Signal and Information Processing*) [137],coordenado pelo professor Joe Picone e localizado na Universidade do Estado do Mississipi (MSU - *Mississippi State University*), é disponibilizar um sistema de reconhecimento de voz que implementa o estado da arte, de forma modular e sem custos de licença, o qual pode ser facilmente modificado para satisfazer a necessidades de pesquisadores. O sistema é construído sobre uma vasta hierarquia de classes de propósito geral escritas em C++, e que implementam conceitos matemáticos, de processamento de sinais e estruturas de dados necessárias ao desenvolvimento de sistemas.

O Sphinx [92] foi desenvolvido na Universidade Carnegie Mellon (CMU - *Carnegie Mellon University*) juntamente com os projetos Sphinx fundados pelo DARPA e criados de forma a estimular a criação de ferramentas e aplicativos de processamento de voz e fazer avançar o estado da arte em reconhecimento de fala, bem como, nas áreas correlatas, como sistemas de diálogo e síntese de voz. Os termos de licença para as ferramentas e bibliotecas disponíveis no Sphinx são derivados do modelo de distribuição de software da Berkeley (BSD - Berkeley Software Distribution). Não há restrição alguma quanto ao uso comercial ou redistribuição. O Sphinx está na versão 4, lançada em 04/06/2004, e implementa o estado da arte para sistemas de reconhecimento de fala baseados em HMMs contínuos. A biblioteca possui um excelente tutorial on-line e sua arquitetura é descrita num documento intitulado "*Sphinx-4 Whitepaper*". Da versão 3 para a versão atual, o Sphinx foi completamente rescrito e portado para a plataforma de linguagem de programação JAVA®.

Neste trabalho, foi utilizado o *framework* Sphinx. A escolha se deu, devido a seu código ser aberto, o que permite a modificação dos parâmetros utilizados, além de ser amplamente referenciado na área [28, 30, 122, 126].

O *framework* Sphinx foi desenvolvido objetivando modularidade e flexibilidade. Isso implica que cada módulo pode ser substituído, de modo a implementar a técnica desejada, sem que isso comprometa o funcionamento do restante do sistema. A arquitetura do Sphinx é apresentada na Figura 5.2.

Os três principais módulos do Sphinx são: *Front End*, *Decoder* e *Linguist*. O módulo *Front End* recebe um ou mais sinais de voz e os transforma em uma sequência de características (*Features*). O módulo *Linguist* traduz informações dos modelos acústico, linguístico e dicionário em um gráfico de busca (*SearchGraph*). Com base nas características forneci-

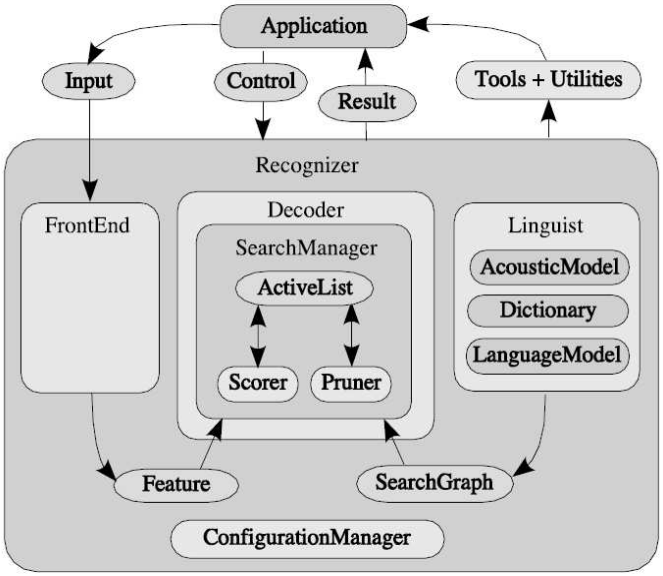


Figura 5.2: *Arquitetura do Sphinx [6].*

das pelo *Front End* e os modelos do *Linguist*, o módulo *Decoder* realiza o reconhecimento, gerando resultados (*Results*).

O Sphinx permite que vários parâmetros sejam configurados, inclusive em tempo de execução, utilizando o módulo *Configuration Manager*. Por exemplo, por padrão o Sphinx utiliza coeficientes MFCC (*Mel-Frequency Cepstral Coefficients*). No entanto, é possível configurar o módulo *Front End*, de modo que sejam calculados, por exemplo, coeficientes PLP (*Perceptual Linear Prediction*).

O *framework* inclui também algumas ferramentas (*Tools*) que permitem obter algumas estatísticas do processo de reconhecimento, como por exemplo, taxa de erro de palavras e sentenças, tempo de execução, quantidade de memória utilizada, etc.

O sub-módulo *LanguageModel*, que faz parte do módulo *Linguist*, provê uma estrutura linguística em nível de palavra (*word-level*). O *Dictionary* fornece a pronúncia das palavras do *LanguageModel*. E o sub-módulo *AcousticModel* realiza o mapeamento entre as unidades de fala e um HMM que pode ser "pontuado" em relação às características fornecidas pelo *FrontEnd*. Essse mapeamento pode levar em consideração informações de contexto e posição na palavra. Por exemplo, no uso de trifones, o contexto representa os fonemas à esquerda e à direita do fonema em questão. A posição na palavra, considera se o fonema está no início, meio ou fim da palavra, ou ainda, se é uma única palavra.

5.3 Base de Dados

As tecnologias mais promissoras na área de reconhecimento de fala, como por exemplo, *Hidden Markov Models* (HMM) utilizam métodos de modelagem estatística que aprendem por exemplos, e devem considerar no treinamento as possíveis variações da fala e o ambiente de utilização do processo do sistema de reconhecimento [91, 131, 133, 138, 139, 140].

O efeito causado por variáveis não modeladas ou mal modeladas (tais como diferenças de canal ou microfones, palavras fora do vocabulário, sub-unidades fonéticas mal treinadas) podem comprometer seriamente o desempenho dos sistemas de reconhecimento de fala. Outro fator que deve ser considerado quando se trata de reconhecimento de fala são as características acústicas e linguísticas de cada idioma. Diferentes idiomas possuem diferentes fonemas, tanto em número quanto em características (duração, intensidade, etc), logo parâmetros e configurações que oferecem altas taxas de reconhecimento em determinado idioma, podem não ser adequados para um outro. Enquanto que, para o idioma inglês existem várias pesquisas e bons resultados, pouco se tem para o idioma português brasileiro [49, 131, 133, 138, 140]. Para um sistema de reconhecimento de fala eficiente, torna-se necessária a adequação dos modelos acústicos e linguísticos para o português brasileiro, além de testes variando os principais parâmetros utilizados [38, 132, 138, 140].

A disponibilidade de bases de dados é fator condicionante para o desenvolvimento de sistemas de Reconhecimento de Fala Contínua. O compartilhamento de bases de dados entre pesquisadores americanos e europeus foram um dos grandes motivos para o progresso obtido nos seus respectivos países, ao longo das últimas décadas. No entanto, para fornecer exemplos em número suficiente para que os métodos estatísticos funcionem adequadamente, a base de dados precisa ser representativa e, consequentemente, custosa, tanto em termos de tempo despendido para sua construção, quanto em termos financeiros. Estes altos custos só podem ser arcados por um esforço conjunto de empresas, instituições de pesquisa e agências financiadoras, de modo a evitar duplicação de esforços, por meio da distribuição de tarefas [38]. Para envolver um maior número de agentes neste processo, é necessário que a base de dados não seja direcionada a um sistema ou tarefa específicos, mas objetive atender às necessidades de vários grupos, com linhas de pesquisa em desenvolvimento, em diversas áreas de aplicação do processamento da fala (síntese e reconhecimento de fala, estudos fonéticos, estudos linguísticos, etc.) [38, 91, 141, 142].

Na Europa, o projeto EUROME_1 congregou esforços de 8 países europeus: Itália, Inglaterra, Alemanha, Holanda, Dinamarca, Suécia, Noruega e França, com a adesão posterior de Grécia, Espanha e Portugal. A base de dados foi criada com o mesmo número de locutores (30 homens e 30 mulheres), escolhidos a partir dos mesmos critérios e gravados em condições acústicas semelhantes, e no mesmo formato.

Em Portugal, foi criada uma base de dados chamada BD-PUBLICO (Base de Dados em Português eUropeu, vocaBulário Largo, Independente do orador e fala COntínua), com aproximadamente 10 milhões de palavras em aproximadamente 156 mil frases, pronunciadas por 120 locutores (60 de cada gênero). Como não poderia deixar de ser, esta base foi confeccionada a partir do esforço conjunto de instituições de pesquisa, órgãos governamentais e também empresas do setor privado.

Nos EUA, também foi feito um grande esforço neste sentido, e já existem disponíveis no domínio público, várias bases de dados para desenvolvimento e teste de sistemas, por exemplo: TIMIT, TI-DIGITS, SWITCHBOARD.

A disponibilidade dessas bases impulsionou de forma expressiva o desenvolvimento da tecnologia de fala, não só devido ao fato dos centros de pesquisa não terem que criar suas próprias bases de dados, um trabalho por si só extremamente árduo, caro e demorado, mas também pela possibilidade de comparar os resultados de cada nova idéia de forma estatisticamente significativa.

Para o português brasileiro, no entanto, ainda há uma grande carência de bases comuns, de forma que apenas iniciativas individuais de centros de pesquisa são eventualmente encontradas [143, 144, 145].

Neste trabalho, foi utilizada a base de dados desenvolvida por Ynoguti e Violaro em [143]. Optou-se pelo uso dessa base, uma vez que sua estrutura é adequada para o teste de aplicações para reconhecimento de fala contínua. A base é formada por diferentes locutores: foram gravadas amostras de voz com 46 locutores, sendo 24 do gênero masculino e 22 do feminino, de diferentes idades (18 a 60 anos), graus de instrução e cidades do Brasil. Além disso, foram utilizadas 200 frases foneticamente balanceadas [142]. Outro motivo que contribuiu para escolha da base foi sua disponibilidade. As frases que formam a base de dados e informações sobre os locutores podem ser encontradas no Apêndice A

A base de dados era composta por:

- 1600 arquivos de áudio no formato WAV com frequência de amostragem de 11 KHz e resolução de 16 bits;
- Respectivos arquivos de transcrição fonética;
- Dicionário fonético (fonemas);
- Dicionário (palavras do vocabulário).

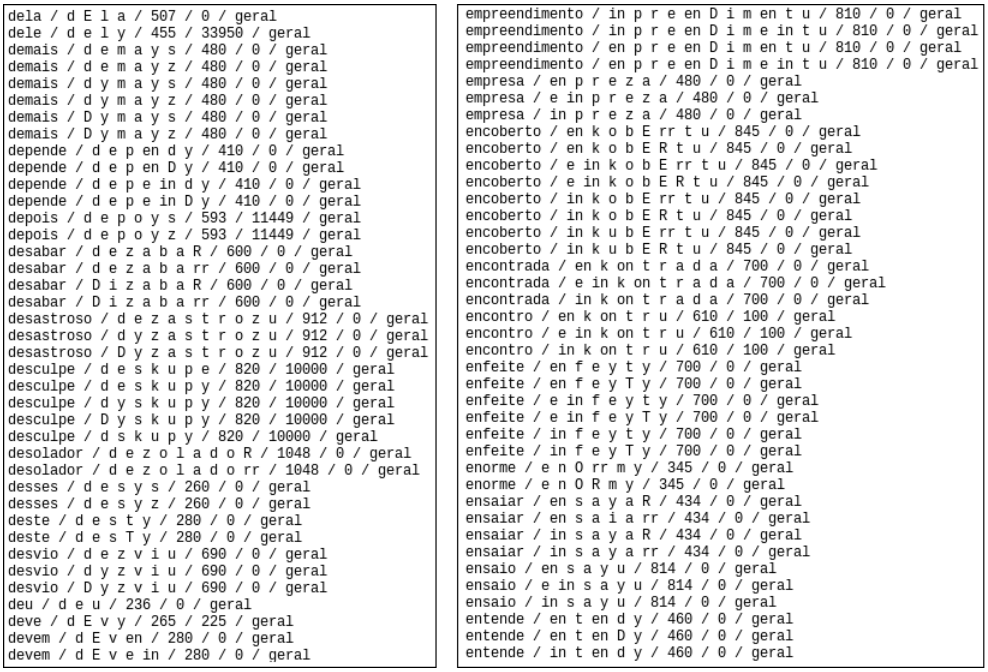
O dicionário consiste de um arquivo contendo todas as palavras do vocabulário com suas respectivas transcrições fonéticas. Na Figura 5.3, é apresentada parte do arquivo de Dicionário.

A transcrição fonética representa a sequência de fonemas que constitui cada palavra. Vale salientar, que uma mesma palavra (transcrição gráfica) pode ter mais de uma transcrição fonética, contemplando assim as variações da pronúncia, ênfase, sotaques locais, entre outros fatores. A estrutura do arquivo de dicionário é a seguinte:

transc. gráfica / transc. fonética / média da duração (ms) / desvio padrão da duração

5.4 Adaptação da Base de Dados

Devido à diferença entre a estrutura dos arquivos da base de dados original e a estrutura utilizada pelo Sphinx, foi necessário realizar algumas adaptações na base de dados para permitir os testes no Sphinx. Tais adaptações serão descritas nesta seção.



```
dela / d E l a / 507 / 0 / geral
dele / d e l y / 455 / 33950 / geral
demais / d e m a y s / 480 / 0 / geral
demais / d e m a y z / 480 / 0 / geral
demais / d y m a y s / 480 / 0 / geral
demais / d y m a y z / 480 / 0 / geral
demais / D y m a y s / 480 / 0 / geral
demais / D y m a y z / 480 / 0 / geral
depende / d e p en d y / 410 / 0 / geral
depende / d e p en D y / 410 / 0 / geral
depende / d e p e in d y / 410 / 0 / geral
depende / d e p e in D y / 410 / 0 / geral
depois / d e p o y s / 593 / 11449 / geral
depois / d e p o y z / 593 / 11449 / geral
desabar / d e z a b a R / 600 / 0 / geral
desabar / d e z a b a rr / 600 / 0 / geral
desabar / D i z a b a R / 600 / 0 / geral
desabar / D i z a b a rr / 600 / 0 / geral
desastroso / d e z a s t r o z u / 912 / 0 / geral
desastroso / d y z a s t r o z u / 912 / 0 / geral
desastroso / D y z a s t r o z u / 912 / 0 / geral
desculpe / d e s k u p e / 820 / 10000 / geral
desculpe / d e s k u p y / 820 / 10000 / geral
desculpe / d y s k u p y / 820 / 10000 / geral
desculpe / D y s k u p y / 820 / 10000 / geral
desculpe / d s k u p y / 820 / 10000 / geral
desolador / d e z o l a d o R / 1048 / 0 / geral
desolador / d e z o l a d o rr / 1048 / 0 / geral
desses / d e s y s / 260 / 0 / geral
desses / d e s y z / 260 / 0 / geral
deste / d e s t y / 280 / 0 / geral
deste / d e s T y / 280 / 0 / geral
desvio / d e z v i u / 690 / 0 / geral
desvio / d y z v i u / 690 / 0 / geral
desvio / D y z v i u / 690 / 0 / geral
deu / d e u / 236 / 0 / geral
deve / d E v y / 265 / 225 / geral
devem / d E v en / 280 / 0 / geral
devem / d E v e in / 280 / 0 / geral
empreendimento / in p r e en D i m en t u / 810 / 0 / geral
empreendimento / in p r e en D i m e in t u / 810 / 0 / geral
empreendimento / en p r e en D i m en t u / 810 / 0 / geral
empreendimento / en p r e en D i m e in t u / 810 / 0 / geral
empresa / en p r e z a / 480 / 0 / geral
empresa / e in p r e z a / 480 / 0 / geral
empresa / in p r e z a / 480 / 0 / geral
encoberto / en k o b E rr t u / 845 / 0 / geral
encoberto / en k o b E R t u / 845 / 0 / geral
encoberto / e in k o b E rr t u / 845 / 0 / geral
encoberto / e in k o b E R t u / 845 / 0 / geral
encoberto / in k o b E rr t u / 845 / 0 / geral
encoberto / in k o b E R t u / 845 / 0 / geral
encoberto / in k u b E rr t u / 845 / 0 / geral
encoberto / in k u b E R t u / 845 / 0 / geral
encontrada / en k on t r a d a / 700 / 0 / geral
encontrada / e in k on t r a d a / 700 / 0 / geral
encontrada / in k on t r a d a / 700 / 0 / geral
encontro / en k on t r u / 610 / 100 / geral
encontro / e in k on t r u / 610 / 100 / geral
encontro / in k on t r u / 610 / 100 / geral
enfeite / en f e y t y / 700 / 0 / geral
enfeite / en f e y T y / 700 / 0 / geral
enfeite / e in f e y t y / 700 / 0 / geral
enfeite / e in f e y T y / 700 / 0 / geral
enfeite / in f e y t y / 700 / 0 / geral
enfeite / in f e y T y / 700 / 0 / geral
enorme / e n O rr m y / 345 / 0 / geral
enorme / e n O R m y / 345 / 0 / geral
ensaiar / en s a y a R / 434 / 0 / geral
ensaiar / en s a i a rr / 434 / 0 / geral
ensaiar / in s a y a R / 434 / 0 / geral
ensaiar / in s a y a rr / 434 / 0 / geral
ensaio / en s a y u / 814 / 0 / geral
ensaio / e in s a y u / 814 / 0 / geral
ensaio / in s a y u / 814 / 0 / geral
entende / en t en d y / 460 / 0 / geral
entende / en t en D y / 460 / 0 / geral
entende / in t en d y / 460 / 0 / geral
```

Figura 5.3: *Formato do dicionário original.*

O Sphinx possui um módulo específico para a realização da etapa de treinamento, o *Sphinx-3 Trainer*. Para realizar o treinamento, é necessário fornecer os seguintes arquivos:

1. Áudio para treinamento;
2. Transcrição fonética dos respectivos arquivos de áudio;
3. Dicionário;
4. Fonemas;
5. Fonemas inter-palavras (*Filler*);
6. Modelo Linguístico.

O Dicionário consiste no vocabulário do sistema, sendo que cada palavra pode ter mais de uma transcrição fonética. A transcrição é realizada com base nos fonemas e nos fonemas inter-palavras. Esses últimos correspondem a fonemas que não representam sons da fala, por exemplo, silêncio no início e fim de sentenças, ou ainda, as pausas entre as palavras da mesma sentença.

Para realização dos testes com a base de dados no Sphinx, foram necessárias as seguintes adaptações:

1. Formato dos arquivos de transcrição;
2. Tratamento de diferentes transcrições para mesma palavra no Dicionário;
3. Tratamento de diferentes fonemas utilizando o mesmo caractere;
4. Construção do Modelo Linguístico.

5.4.1 Arquivos de transcrição

Ao contrário da base de dados original, que utiliza um arquivo de transcrição para cada arquivo de áudio, no Sphinx existe um único arquivo no qual estão todas as transcrições. Nesse arquivo, não são utilizados caracteres de pontuação (ponto, vírgula, exclamação, etc), cada linha corresponde a um arquivo de áudio, cujo nome é informado no final da linha e entre parênteses.

Também foi necessário sinalizar os intervalos de silêncio no início, fim e entre as palavras de uma sentença. Na Figura 5.4, é ilustrada parte do arquivo de transcrição gerado no formato do Sphinx.

```
<s> o k l i m a n a n u E m a u e ĩ n k a u k u t a </s> (F020808)
<s> a l o k o m o T i v a v e i n s e i n m u i n t a k a R g a </s> (F020809)
<s> a i n d E u m a b o u a t e n p o r a d a p a r o s i n e m a </s> (F020810)
<s> u z m a y O y s p i k u z d a t E r r a f i k a n u d e b a y x u d a g u a </s> (F030901)
<s> a i n a u g u r a s a n u d a v i l a E k u a R t a f e y r a </s> (F030902)
<s> s O v O t a k e i n T i v E r u T i t u l u D y e l e y t o R </s> (F030903)
<s> E f u n d a m e n t a u b u s k a r a r r a z a n u d a e z i s t e i n s y a </s> (F030904)
<s> a t e n p e r a t u r a s O E b o a m a y s e d u </s> (F030905)
<s> e i n m u y t a z r r e j i o n y z a p o p u l a s a n u y s t a D i m i n u i n d u </s> (F030906)
<s> n u n k a s y p o D y f i k a r i n s i m a d u m u r u </s> (F030907)
<s> p r a k e i n v e D y f O r a o p a n n o r a n m a E D i z o l a d o R </s> (F030908)
<s> E b o u n T y v e R f <sil> k o L e n d u f l o r y s </s> (F030909)
<s> e u m y b a n N u n u l a g u a u a m a n N e s e </s> (F030910)
<s> E f u n d a m e n t a u x e g a r a u n m a s o l u s a n u k o m u n </s> (F031001)
<s> a p r e v i z a n u D y m u y t u n e v o e y r u n u r r i u </s> (F031002)
<s> m u i n t u s m O v e y z v i r a n u a s <sil> s i n g u d a t a R D y </s> (F031003)
<s> a k a z a p O D y D y z a b a r i n a u g u m a z O r a s </s> (F031004)
<s> u k a n D i d a t u f a l o u k o m u s y e s T i v E s e l e y t u </s> (F031005)
<s> a i D E y E f a L a m a z i n t e r E s a <sil> </s> (F031006)
<s> o D i a y s t a b o n p a r a p a s y a n o k i n t a u </s> (F031007)
<s> m i n N a s k o r r e s p o n d e i n s y a z n a n u y s t a n u i n k a z a </s> (F031008)
<s> a s a i d a p a r a a k r i s y d e l y E u D i a l u g u </s> (F031009)
<s> f i n a u m e i n T y u m a u t e n p u d e y x o u k o n t i n e i n T y </s> (F031010)
<s> u n k a z a u D y g a t u s k O m y n u t e L a d u </s> (F031101)
<s> a k a n t o r a f o y a p r e z e n t a R s e u T i m u s u s E s u </s> (F031102)
<s> l a E u n l u g a r O T i m u p a r a t o m a r u n s x o p i n N u s </s> (F031103)
<s> o m u z i k a u k u n s u m y u s E T y m e z y z D y e n s a y u </s> (F031104)
<s> n O s u b a y l i n i s i a p O z a z n O v y </s> (F031105)
<s> a p e z a R d e s y s r r e z u t a d u s <sil> t o m a r e y u n m a d e s i z a n u </s> (F031106)
<s> a v e R d a D y n a n u p o u p a n e i n a s e l e b r i d a D y s </s> (F031107)
<s> a s k e y m a d a z d E v e i n D i m i n u i r e s T y a n n u </s> (F031108)
<s> u v a n u e n t r y u t r e i n y a p l a t a f O R m a E m u y t u g r a n D y </s> (F031109)
<s> i n f e l i z m e i n T y n a n u k o n p a r e s i a u e i n k o n t r u </s> (F031110)
```

Figura 5.4: *Formato do arquivo de transcrição gerado.*

5.4.2 Diferentes transcrições

Algumas palavras possuem mais de uma transcrição fonética. Isso pode ocorrer devido a diversos fatores: sotaque regional, velocidade da pronúncia, fusão com fonemas de palavras anteriores ou posteriores, devido ao efeito da coarticulação, dentre outros.

Para que essas diferentes pronúncias possam ser tratadas pelo sistema, isso deve estar registrado no Dicionário. No Sphinx, isso é informado numerando-se a partir da segunda ocorrência da palavra. Na Figura 5.5, é apresentada parte do Dicionário nos quais são tratadas as diferentes pronúncias das palavras “celebridades” e “certas”.

É possível observar na Figura 5.5 as diferentes transcrições fonéticas (variações na pronúncia) para as palavras destacadas.

```
cantora k an t o r a
capital k a p i t a u
captado k a p t a d u
captado(2)      k a p i t a d u
carga   k a rr g a
carga(2)        k a rrr g a
casa    k a z a
casal   k a z a u
casamento       k a z a m en t u
casamento(2)    k a z a m e in t u
causa   k a u z a
cá      k a
cedo    s e d u
celebridades    s e l e b r i d a d s
celebridades(2) s e l e b r i d a dd y s
celebridades(3) s e l e b r i d a d y s
celebridades(4) s e l e b r i d a d z
celebridades(5) s e l e b r i d a dd y z
celebridades(6) s e l e b r i d a d y z
cenário s e n a r y u
certas  s ee rr t a s
certas(2)       s ee rrr t a s
certas(3)       s ee rr t a z
certas(4)       s ee rrr t a z
certo   s ee rr t u
certo(2)        s ee rrr t u
chance  x an s y
chance(2)       x an s
chegar  x e g a rrr
chegar(2)       x e g a rr
chopinhos       x o p in nn u s
chopinhos(2)    x o p in nn u z
cidade  s i d a d y
cidade(2)       s i d a dd y
cima    s i m a
cinco   s in k u
cinema  s i n e m a
classe  k l a s y
clima   k l i m a
```

Figura 5.5: *Formato do dicionário modificado.*

5.4.3 Diferentes fonemas

No arquivo de fonemas são armazenados os fonemas utilizados na transcrição fonética das palavras do Dicionário e das sentenças dos arquivos de áudio. No arquivo original da base de dados utilizada, em alguns casos, o mesmo caractere era utilizado para representar diferentes tons do mesmo fonema, variando apenas entre maiúsculo e minúsculo, para tons de maior e menor intensidade, respectivamente. Por exemplo:

Fonema	Representação	Exemplo
/e/	e	pêssego
/é/	E	berro, café

Contudo, o Sphinx não faz distinção entre caracteres maiúsculos e minúsculos, e assim, os diferentes fonemas representados pelo mesmo caractere, variando apenas o *case*, eram considerados duplicações no arquivo de Fonemas e era gerado um erro durante o início da etapa de treinamento. Consequentemente, isso também propagava erros no Dicionário e no arquivo de transcrição, uma vez que, os dois utilizam os fonemas definidos em tal arquivo. Para resolver essa situação, foi adotada a seguinte estratégia: em fonemas representados por um caractere já utilizado (mudando apenas o *case*), o fonema mais forte passaria a ser representado pelo mesmo caractere, só que duplicado. Caso essa representação já existisse,

era acrescentado mais um caractere, que agora seria triplicado. Tal estratégia é escalável, permitindo o tratamento de novos fonemas. Outra exigência do Sphinx é que o fonema referente ao silêncio esteja presente neste arquivo. Na Figura 5.6, são apresentados os arquivos de Fonemas original e modificado. Na Figura também é destacado o caso especial no qual o caractere precisou ser triplicado, uma vez que, o fonema "rr" já existia.

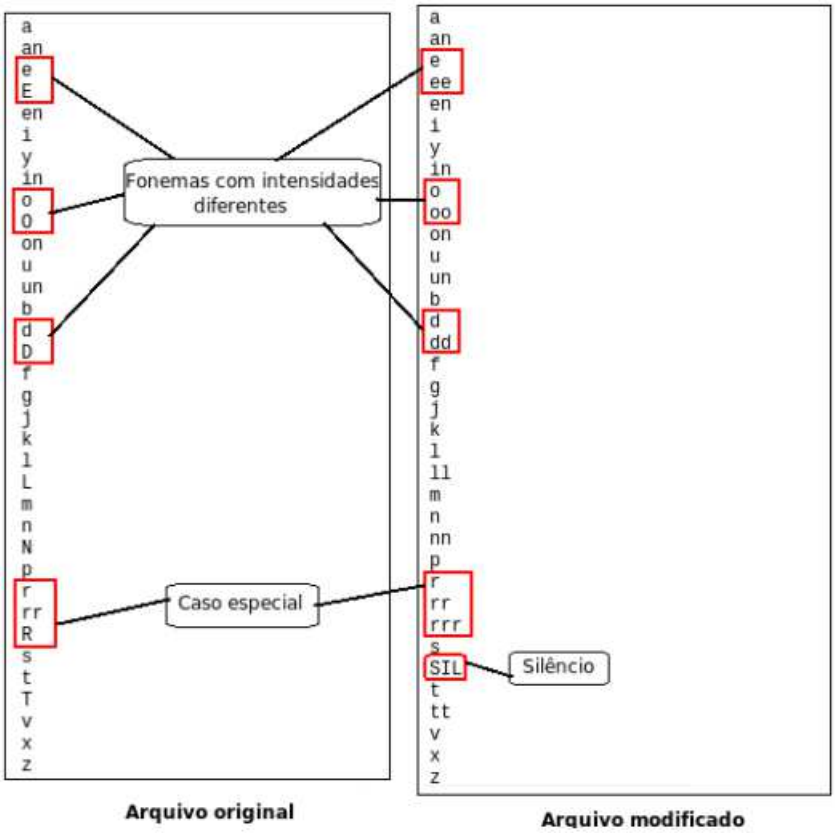


Figura 5.6: *Arquivo de Fonemas original e modificado.*

É importante ressaltar, que a modificação dos fonemas teve que ser repassada para o Dicionário e o arquivo de transcrição.

5.4.4 Construção do Modelo linguístico

Um outro elemento fundamental para realização de reconhecimento de fala contínua é o Modelo Linguístico (ML). A base de dados original não oferecia um ML, o qual teve que ser gerado.

Para esta tarefa, foi utilizada uma ferramenta *lmtool* do próprio Sphinx disponível em: [146]. Com base no *corpus* de sentenças do sistema, foi construído o ML utilizando o modelo N-gram. Na Figura 5.7, são ilustradas as partes iniciais de cada modelo (1-gram, 2-gram, 3-gram) do ML gerado pela ferramenta.

O arquivo gerado pela ferramenta *lmtool* é no formato texto. Contudo, o Sphinx utiliza o padrão ARPA, cujo formato do arquivo é DMP. Para realizar essa conversão, é disponibi-

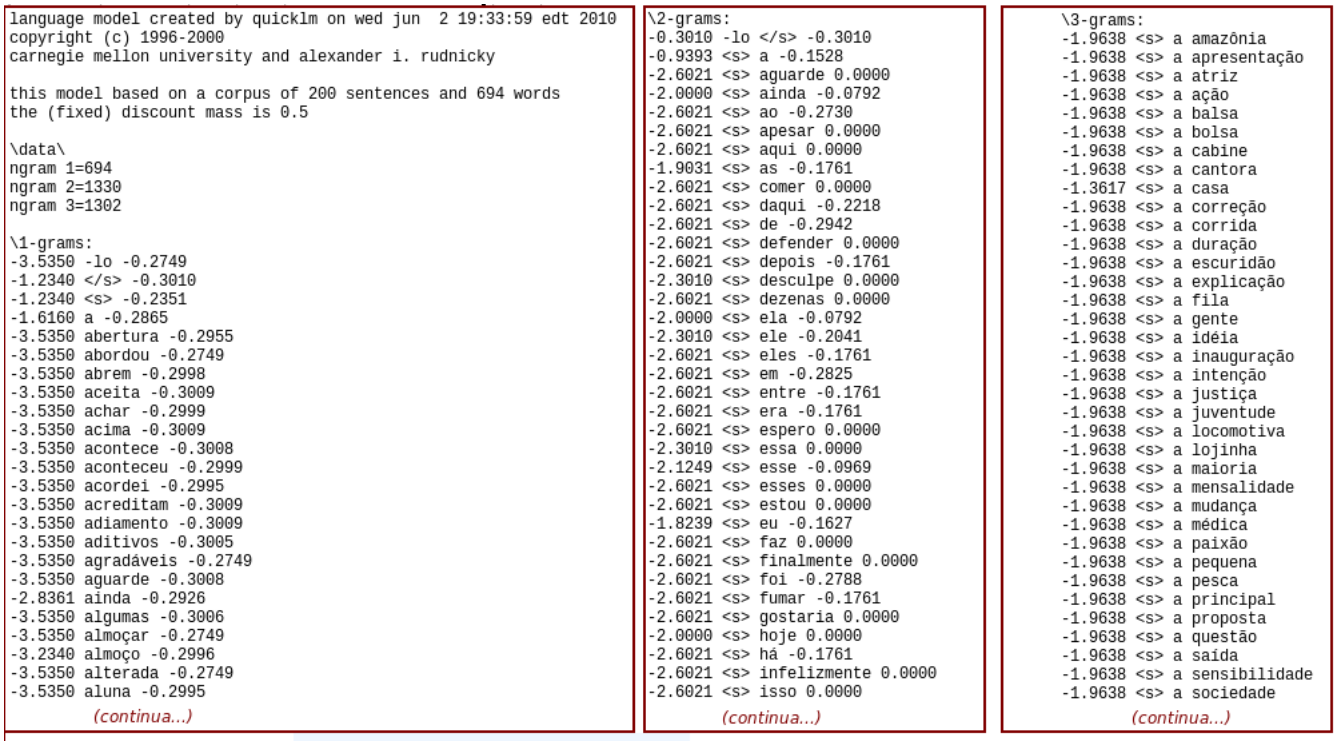


Figura 5.7: *Modelo Linguístico gerado.*

lizada a ferramenta *sphinx_lm_convert*.

5.4.5 Validação das adaptações realizadas

Uma vez realizadas as modificações apresentadas nas seções anteriores, a etapa seguinte consistiu da validação dessas, efetuando-se treinamentos e testes com a nova versão da base de dados.
Para isso, a modelagem utilizada foi:

- HMM contínuos com 3 estados para cada fonema
- 13 coeficientes mel-cepstrais
- 40 filtros mel-cepstrais

Na etapa de treinamento, foram utilizados 1200 arquivos de áudio contendo diferentes ocorrências das 200 sentenças que formam o *corpus* da base de dados, juntamente com os novos arquivos de:

- Transcrição fonética (agora em um único arquivo);
- Dicionário;
- Fonemas;
- Fonemas inter-palavras (*Filler*);

- Modelo Linguístico.

Na etapa de teste, foram utilizados 40 arquivos de áudio obtendo uma taxa de 92,8% de reconhecimento para palavras e 80% para sentenças. Diante desses resultados, pode-se considerar que as adequações efetuadas foram validadas.

5.5 Cenários de Testes

Como citado anteriormente, neste trabalho foi realizada uma investigação da configuração e parâmetros mais adequados para modelagem acústica do sinal de voz, considerando o português brasileiro e as limitações do contexto embutido. Os testes foram realizados utilizando o *framework* Sphinx3 em uma máquina com processador AMD Turion X2 de 800 MHz e 2 GB de memória principal. Como foi descrito na Seção 5.3, a base de dados utilizada é composta por 200 frases foneticamente balanceadas, pronunciadas por 46 diferentes locutores. As frases que formam a base de dados e informações sobre os locutores podem ser encontradas no Apêndice A.

Os cenários dos testes realizados são descritos a seguir.

- 1200 sentenças para treinamento e 400 para reconhecimento;
- 19 locutores masculinos e 17 femininos para treinamento;
- 05 locutores de cada gênero, totalizando 10 locutores para reconhecimento;
- HMM com 3 e 5 estados;
- de 8 a 10 coeficientes mel-cepstrais;
- banco de filtros na escala mel variando de 20 a 40 filtros;

Uma vez que o objetivo deste trabalho é identificar uma configuração que forneça boas taxas de reconhecimento sem exigir muitos recursos computacionais, embora a literatura aponte para o uso de 12 ou 13 coeficientes MFCC [34, 131, 132, 133, 134], foram realizados testes com outros valores abaixo desse valor. O intuito desses testes é avaliar o impacto da utilização de um número reduzido de coeficientes na taxa de reconhecimento, além do tempo de execução para as etapas de treinamento e reconhecimento diante da variação do número de coeficientes. Vale salientar, que esse tempo é relativo em função da configuração da máquina na qual foram realizadas as simulações. Uma métrica mais precisa que poderia ser utilizada seria a quantidade de ciclos de *clock* do processador, ou ainda o número de operações lógicas e aritméticas necessárias. Contudo, uma vez que todas as simulações foram realizadas em um mesmo cenário, ou seja, na mesma máquina, é possível se ter uma visão normalizada da sobrecarga computacional para execução de cada etapa. Também vale destacar, que os

locutores da etapa de treinamento são diferentes daqueles da etapa de reconhecimento, o que garante o reconhecimento independente de locutor.

Os gráficos apresentados na próxima seção, permitirão a análise das relações entre:

- Número de coeficientes e Taxa de reconhecimento;
- Número de coeficientes e Tempo de treinamento e reconhecimento;
- Número de filtros e Taxa de Reconhecimento;
- Número de filtros e Tempo de treinamento e reconhecimento.

É importante observar, que nos gráficos serão informadas as taxas de reconhecimento para sentenças completas e para palavras isoladas. Em alguns casos, uma sentença pode não ser reconhecida completamente mas, apenas algumas das palavras que a compõem. Por exemplo, na sentença “Muito prazer em conhecê -lo”, que é composta por cinco palavras, podem ser reconhecidas “muito”e “prazer”, e as demais não. Tal situação, produziria uma taxa de reconhecimento de 40% das palavras e 0% de sentenças.

5.6 Resultados dos testes para número de Estados e de Coeficientes

Para facilitar a visualização dos resultados, esses serão apresentados separados por número de estados do HMM.

5.6.1 HMM com 3 estados

Na Figura 5.8, é exibida a taxa de reconhecimento de sentenças em função do número de filtros utilizados para cálculo dos coeficientes mel cepstrais.

Na Figura 5.9, é exibida a taxa de reconhecimento de palavras em função do número de filtros.

Foram realizadas medições do tempo de treinamento em função da variação do número de filtros e coeficientes. O número de filtros praticamente não influenciou no tempo de treinamento e reconhecimento, contudo o número de coeficientes sim. Na Figura 5.10, é exibida a relação entre o tempo de reconhecimento das 400 sentenças em função do número de filtros.

Na Figura 5.11 é apresentado o gráfico que mostra a média dos tempos de treinamento (1200 sentenças) e reconhecimento (400 sentenças) em função do número de coeficientes.

Como pode ser observado na Figura 5.11, a quantidade de coeficientes influencia diretamente no tempo de treinamento e reconhecimento. Quanto menor o número de coeficientes, menor o tempo de treinamento, contudo, o tempo necessário para reconhecimento aumenta.

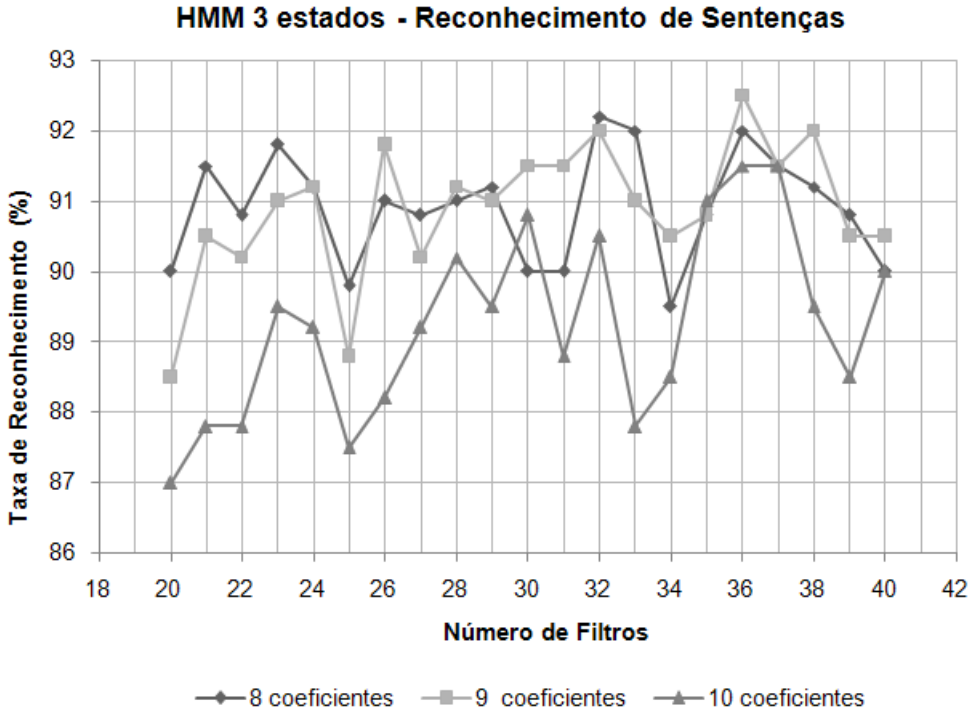


Figura 5.8: *Relação taxa de reconhecimento de sentenças versus número de filtros para HMM de 3 estados, em função do número de coeficientes.*

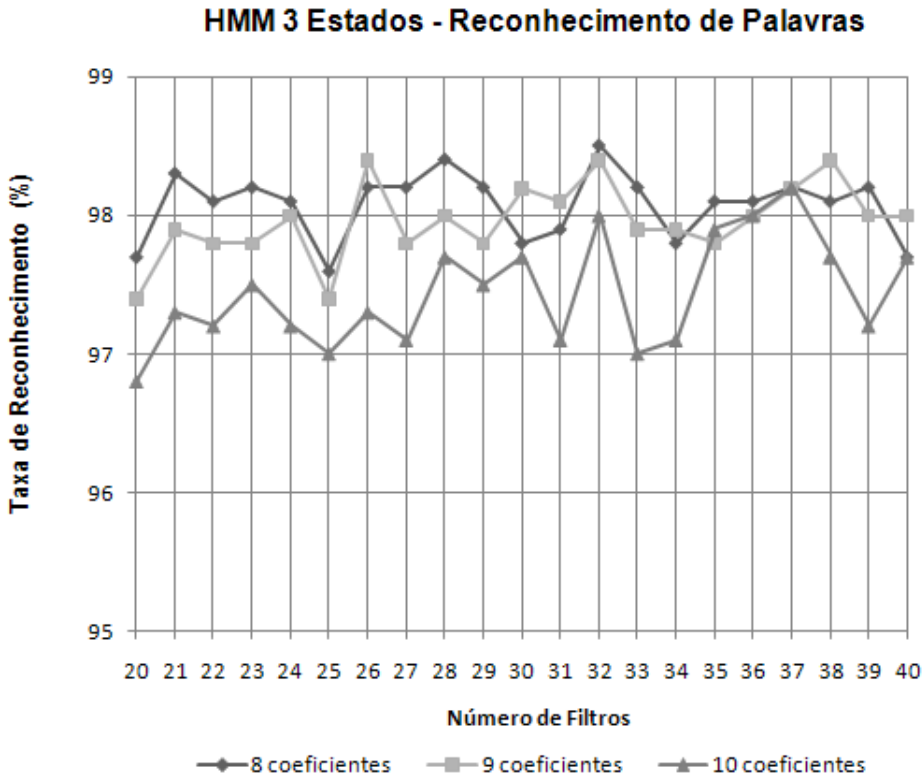


Figura 5.9: *Relação taxa de reconhecimento de palavras versus número de filtros para HMM de 3 estados, em função do número de coeficientes.*

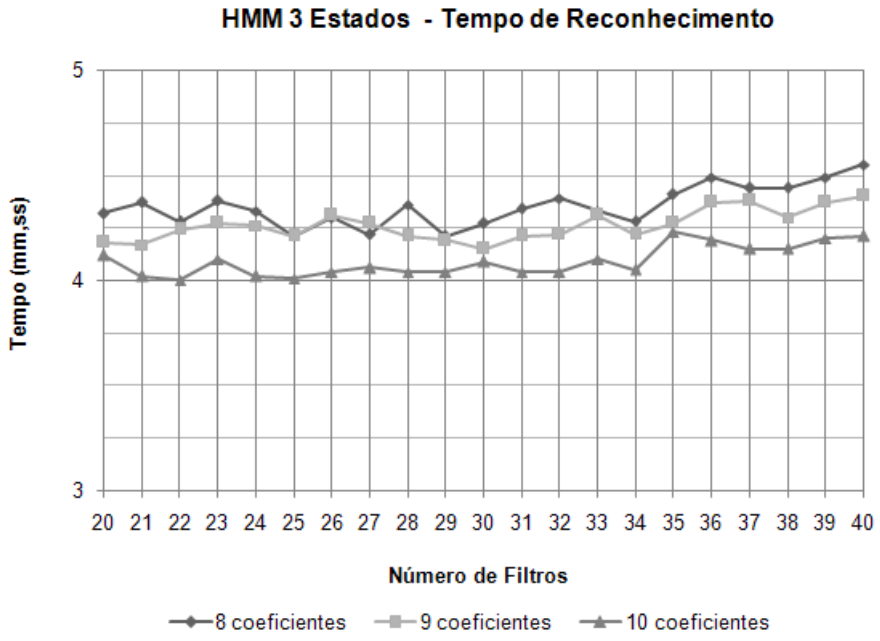


Figura 5.10: *Relação tempo de reconhecimento de palavras versus número de filtros para HMM de 3 estados, em função do número de coeficientes.*

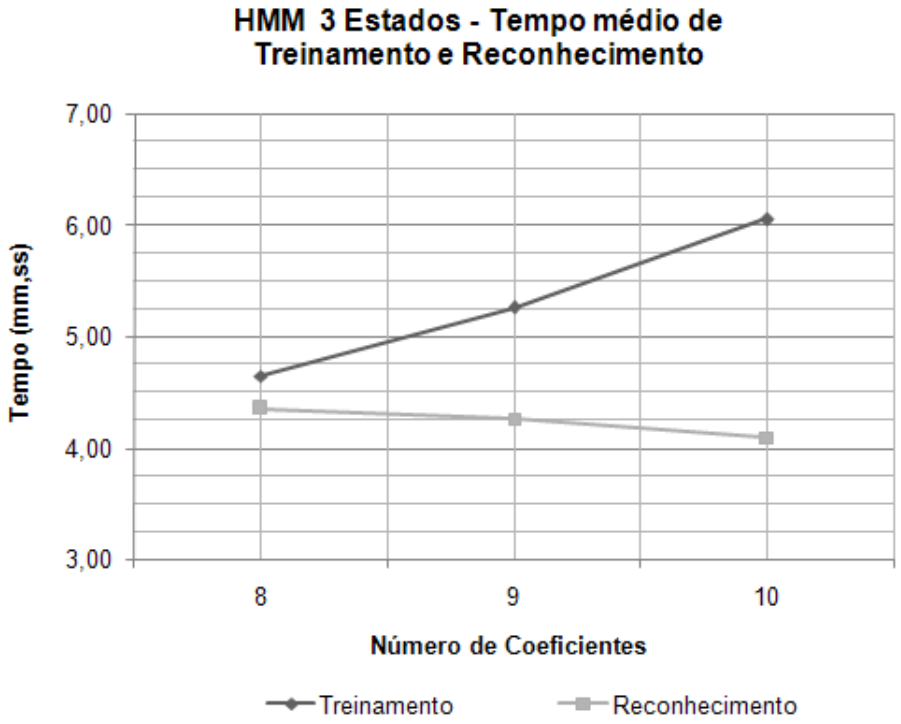


Figura 5.11: *Relação tempos de treinamento (1200 sentenças) e reconhecimento (400 sentenças) versus número de coeficientes para HMM de 3 estados, em função do número de coeficientes.*

5.6.2 HMM com 5 estados

Na Figura 5.12, é exibida a taxa de reconhecimento de sentenças em função do número de filtros utilizados para cálculo dos coeficientes mel cepstrais.

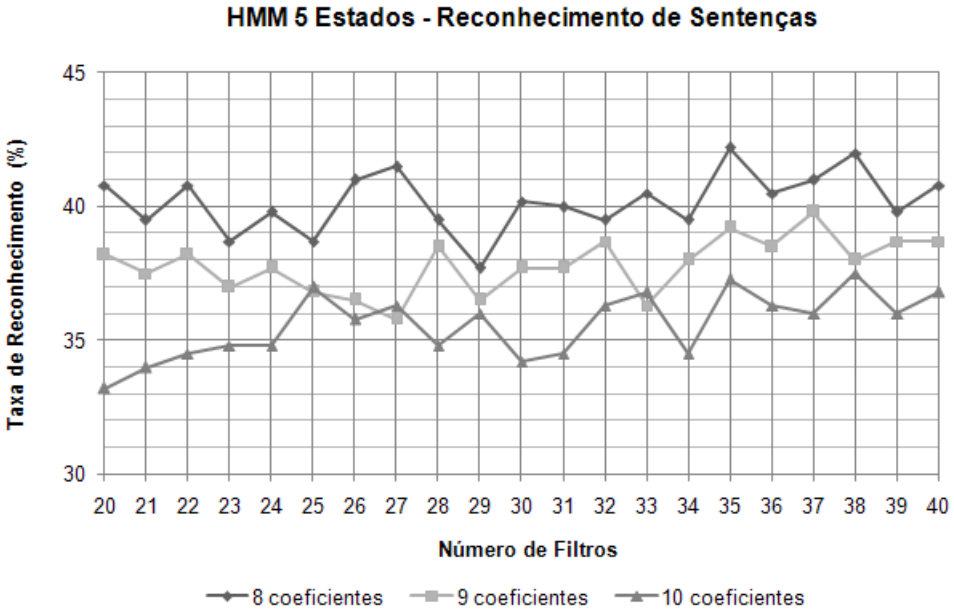


Figura 5.12: *Relação taxa de reconhecimento de sentenças versus número de filtros para HMM de 5 estados, em função do número de coeficientes.*

Na Figura 5.13, é exibida a taxa de reconhecimento de palavras em função do número de filtros.

Foram realizadas medições do tempo de treinamento em função da variação do número de filtros e coeficientes. De forma similar ao HMM de 3 estados, o número de filtros praticamente não influenciou no tempo de treinamento e reconhecimento, contudo, novamente, o número de coeficientes sim.

Na Figura 5.14, é exibida a relação entre o tempo de reconhecimento das 400 sentenças em função do número de filtros.

Na Figura 5.15, é a apresentado o gráfico que mostra a média dos tempos de treinamento (1200 sentenças) e reconhecimento (400 sentenças) em função do número de coeficientes.

Como pode ser observado na Figura 5.15, da mesma forma que no HMM de 3 estados, a quantidade de coeficientes influencia diretamente no tempo de treinamento e reconhecimento. Quanto menor o número de coeficientes, menor o tempo de treinamento, contudo, o tempo necessário para reconhecimento aumenta.

5.6.3 Análise dos resultados obtidos

Diante dos gráficos apresentados, é possível observar que as melhores taxas de reconhecimento para sentenças, que é o objetivo deste trabalho, foram obtidas com HMM de 3 estados, 36 filtros e 9 coeficientes. Outro dado importante que foi obtido, refere-se ao tempo

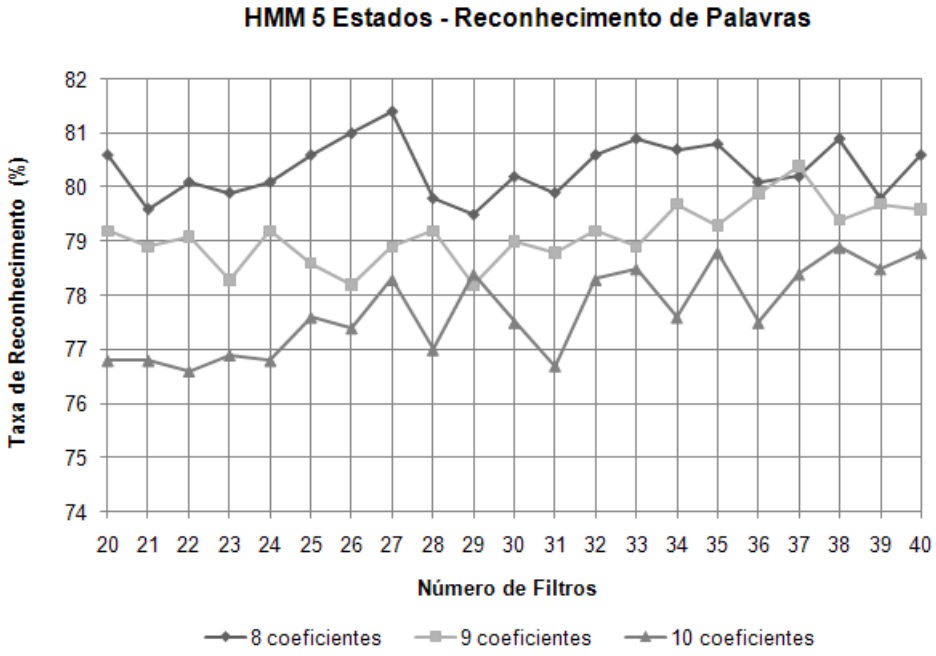


Figura 5.13: *Relação taxa de reconhecimento de palavras versus número de filtros para HMM de 5 estados, em função do número de coeficientes.*

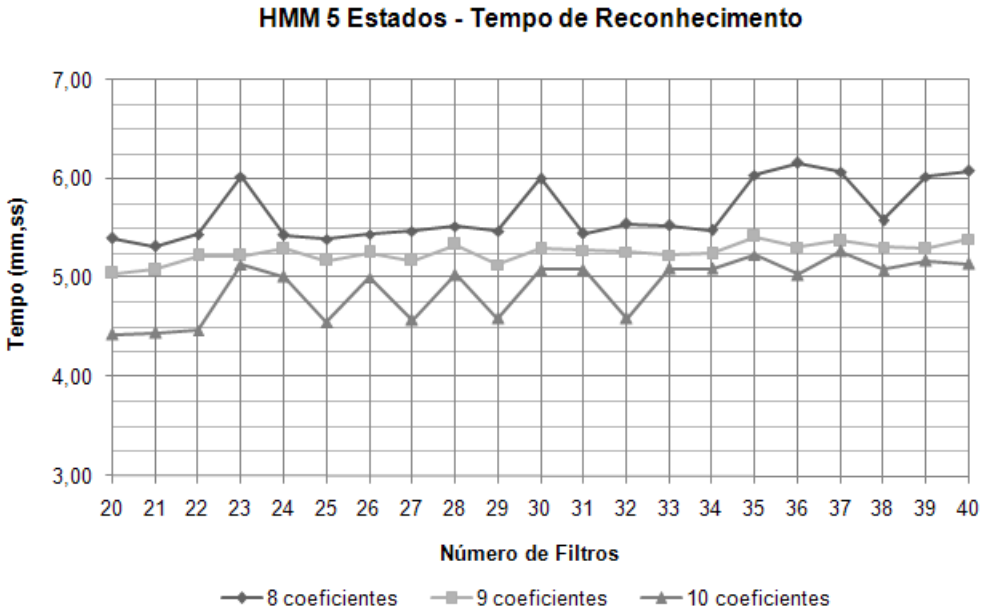


Figura 5.14: *Relação tempo de reconhecimento de palavras versus número de filtros para HMM de 5 estados, em função do número de coeficientes.*

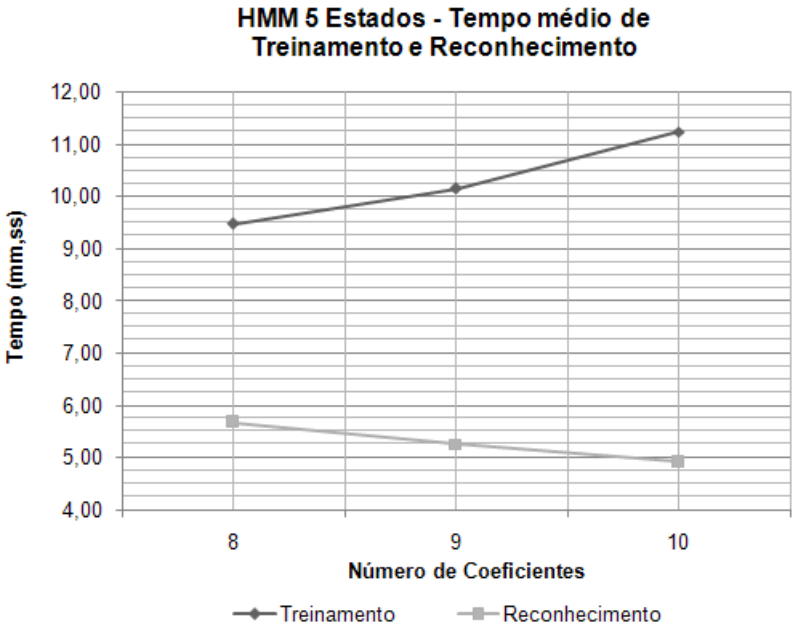


Figura 5.15: *Relação tempos de treinamento (1200 sentenças) e reconhecimento (400 sentenças) versus número de coeficientes para HMM de 5 estados.*

necessário para treinamento e reconhecimento ao variar o número de coeficientes.

Diante dos resultados dos testes realizados, para cada informação obtida, foram tomadas as seguintes decisões em relação à arquitetura do sistema de reconhecimento modelada.

1. Número de estados do HMM

   Analisando-se os gráficos apresentados, é possível concluir que as melhores taxas de reconhecimento foram obtidas com HMM de 3 estados. Sendo esse o número de estados utilizados neste trabalho. Embora um número maior de estados devesse fornecer uma maior taxa de reconhecimento, é importante destacar que o modelo foi treinado para modelar os fonemas e com a propriedade *skipstate* (pular estado) não permitida. Isso pode obrigar o HMM a passar por mais estados do que realmente é necessário para representar determinado fonema, aumentando assim o erro.

2. Número de coeficientes

   Diante dos gráficos apresentados, é possível observar que a quantidade de coeficientes influencia diretamente no tempo de treinamento. Quanto menor o número de coeficientes, menor o tempo de treinamento, contudo, o tempo necessário para reconhecimento aumenta. Levando-se em consideração os resultados dos testes e os aspectos citados, optou-se pela utilização de 9 coeficientes.

3. Número de filtros

   Para o reconhecimento de sentenças, as melhores taxas foram obtidas com 36 filtros.

Como o número de filtros não influenciou consideravelmente no tempo de reconhecimento, e outros trabalhos apontam para esse valor [147, 148, 149, 150, 151], optou-se por utilizar 36 filtros.

4. Utilização de uma memória externa para gravação dos modelos

   Analisando o tempo necessário para treinamento dos modelos, e considerando alguns trabalhos relacionados [27, 33, 34, 119, 120, 126, 129], esta etapa será realizada de maneira *offline* via *software*, uma vez que requer maior capacidade de processamento. Os modelos acústico e linguístico serão gravados em uma memória externa sendo utilizados no momento do reconhecimento. Essa estratégia também se torna interessante porque, com essa configuração, é possível a adaptação do sistema a novos padrões, bastando que a memória onde ficam armazenados os modelos seja atualizada com os novos modelos acústico e linguístico.

5.7 Análise do Cálculo das Gaussianas

Depois de extraídas as características do sinal de voz, a próxima etapa do processo de reconhecimento da fala, consiste em identificar a qual padrão treinado estas correspondem. Para isso, o primeiro passo é calcular a probabilidade que essas caracteristicas tenham sido geradas por cada um dos padrões conhecidos. Para um sistema que utiliza HMM contínuo, como foi exposto no Capítulo 3, a probabilidade dos modelos acústicos é expressa por uma mistura finita de M gaussianas multidimensionais, dada por:

$$p_j = \sum_{m=1}^{M} \frac{w_{j,m}}{2\pi \prod_{k=1}^{K} \sigma_{j,m,k}} exp\left(\sum_{k=1}^{K} \frac{(x_k - \mu_{j,m,k})^2}{2\sigma_{j,m,k}^2} \right), \tag{5.1}$$

sendo j o índice do modelo acústico atual e M o número de misturas utilizadas no modelo, que neste trabalho corresponde a 8 misturas gaussianas. x_k são os coeficientes MFCC, juntamente com suas primeira e segunda derivadas. Como pode ser observado na Equação 5.1, os modelos acústicos são definidos por suas médias (μ), variância (σ^2) e coeficientes de peso de cada mistura (w).

Algumas pesquisas mostram que esta etapa é responsável por cerca de 75% do processo de decodificação [28, 29, 33, 34, 128], e por isso, é objeto de estudo na busca de simplificações e otimizações que facilitem sua implementação em sistemas embarcados de baixo custo. Ao analisar a Equação 5.1, é possível verificar que sua implementação é complexa e envolve um número elevado de operações não lineares. Contudo, foi provado que utilizando o valor máximo de todas as misturas ao invés da soma, é obtida uma boa aproximação do resultado [34, 152]. Sendo assim, usando o negativo do logaritmo da Equação 5.1, de modo a converter probabilidades em custos e aplicando a aproximação citada, o custo para cada

modelo acústico pode ser calculado por:

$$c_j = \min_{m=1}^{M} \left(\alpha_{j,m} + \sum_{k=1}^{K} \frac{(x_k - \mu_{j,m,k})^2}{2\sigma_{j,m,k}^2} \right) \tag{5.2}$$

sendo α o coeficiente por mistura que compensa todas as constantes e parâmetros fora da função exponencial da Equação 5.1.

De modo a simplificar ainda mais o cálculo do módulo das gaussianas, a variância σ^2 pode ser substituída por uma nova variável v' que é descrita segundo a Equação 5.3. Se este valor for pré-calculado e utilizado ao invés da variância, o cálculo do custo de cada vetor de característica para um determinado modelo acústico, é dado pela Equação 5.4.

$$v' = \frac{1}{\sqrt{2\sigma^2}} \tag{5.3}$$

$$c_j = \min_{m=1}^{M} \left(\alpha_{j,m} + \sum_{k=1}^{K} ((\mu_{j,m,k} - x_k).v'_{j,m,k})^2 \right) \tag{5.4}$$

Para avaliar o impacto destas simplificações no cálculo das gaussianas, foram realizados testes com os modelos gerados durante a etapa de treinamento deste trabalho. Foram utilizadas 8 gaussianas, 1108 modelos e vetor de característica com 13 coeficientes MFCC e suas derivadas. O resultado desta avaliação em número de operações é apresentado na Figura 5.16 e na Tabela 5.1.

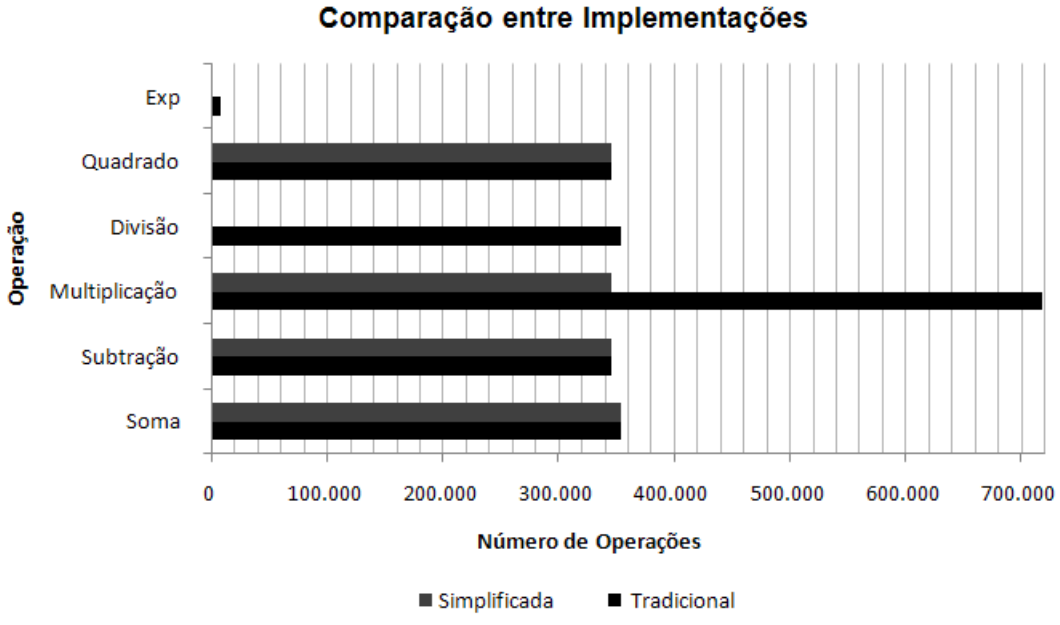


Figura 5.16: *Impacto no número de operações diante da simplificação no Cálculo das Gaussianas.*

Operação	Tradicional	Simplificada
Soma	354.560	354.560
Subtração	345.696	345.696
Multiplicação	717.984	345.696
Divisão	354.560	0
Quadrado	345.696	345.696
Exponencial	8.864	0

Tabela 5.1: *Impacto no número de operações diante da simplificação no Cálculo das Gaussianas.*

Como é possível observar, para as operações de soma e subtração não houve redução considerável. Contudo para operações que possuem um custo computacional mais elevado como multiplicação e exponenciais, houve uma redução de 50% e até mesmo 100%, como é apresentado na Figura 5.17. Nos testes realizados também foi analisado o erro das simplificações efetuadas, que foi de aproximadamente 1%.

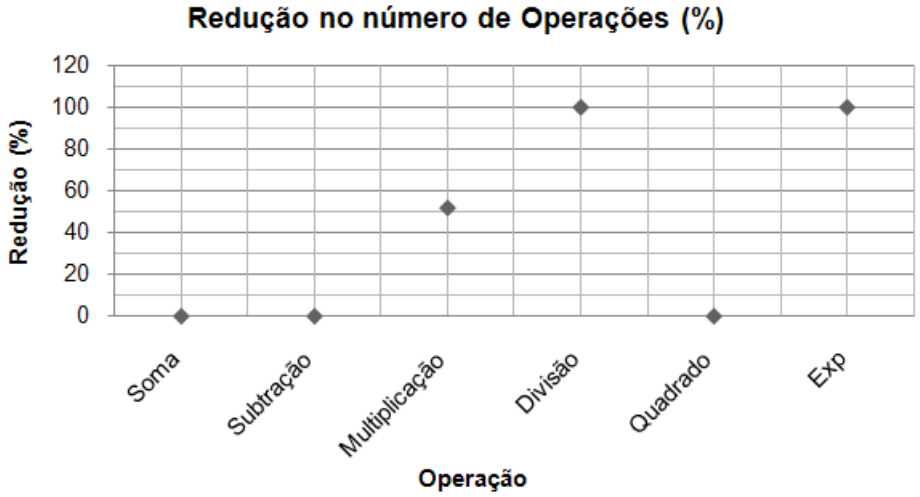


Figura 5.17: *Redução no número de operações diante da simplificação no Cálculo das Gaussianas.*

5.8 Análise da Etapa de Decodificação

A decodificação em um sistema utilizando HMM, consiste na determinação da sequência de estados $q_i = (q_1, q_2, ..., q_T)$ do modelo λ, que mais provavelmente produziu a sequência de observações $O = (0_1, o_2, ..., o_T)$. Como foi explicado no Capítulo 3, por definição, em um HMM a sequência de estados é escondida. Existe apenas a possibilidade de reproduzir a sequência de estados que tem a mais alta probabilidade de ter gerado a sequência de observações dada. Uma das soluções para resolução deste problema é o Algoritmo de Viterbi.

Para determinar a sequência de estados ótima, é necessário calcular a máxima probabilidade, ao longo de um caminho, que termina no estado i, estando no tempo t e produzindo as primeiras t observações. Definindo-se a variável $\delta_t(i)$ como o maior valor de probabilidade ao longo de um único caminho até o instante de tempo t, ou seja, considerando-se as t primeiras

observações que terminam no estado q_i, tem-se por indução que:

$$\delta_{t+1}(j) = [\underset{i}{max}\delta_t(i)a_{ij}]b_j(O_{t+1}) \quad , 1 \leq i \leq N. \tag{5.5}$$

Para se obter a sequência de estados, é necessário reter a trilha do argumento que maximiza a Equação 5.5, para cada t e j. Para tanto, define-se a variável $\psi_t(j)$. O método para se encontrar a sequência de estados ótima é dado por [50, 87]:

1. Inicialização:

$$\begin{aligned} \delta_1(i) = \pi_i b_i(O_1) \quad , 1 \leq i \leq N, \\ \psi_1(i) = 0. \end{aligned} \tag{5.6}$$

2. Recursividade:

$$\begin{aligned} \delta_t(j) = \underset{1\leq i\leq N}{max}[\delta_{t-1}(i)a_{ij}]b_j(O_t), \\ \psi_t(j) = \underset{1\leq i\leq N}{argmax}[\delta_{t-1}(i)a_{ij}], \quad 2 \leq t \leq T, \quad 1 \leq j \leq N. \end{aligned} \tag{5.7}$$

3. Término:

$$\begin{aligned} P^* = \underset{1\leq i\leq N}{max}[\delta_T(i)], \\ q_T^* = \underset{1\leq i\leq N}{argmax}[\delta_T(i)]. \end{aligned} \tag{5.8}$$

4. Sequência de estados ótima

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \quad , t = T-1, T-2, \ldots, 1. \tag{5.9}$$

O algoritmo descrito tem, portanto, a propriedade de determinar a sequência de estados, que maximiza a probabilidade $P(O^l|\lambda_l)$, para a $l-$ésima palavra [5, 87].

A probabilidade de uma palavra (ou sentença) tenderá a zero durante a utilização do algoritmo de Viterbi. Para um número grande de quadros de análise, esta probabilidade irá ultrapassar a precisão dos computadores digitais, resultando em *underflow* [15, 31, 32]. Consequentemente, é necessária a realização de algum tipo de escalonamento. Um caminho para se evitar *underflow* é representar probabilidades pelos seus logaritmos [15, 31, 32]. Utilizando o logaritmo dos parâmetros do modelo no algoritmo de Viterbi, para o cálculo da verossimilhança, não há necessidade de normalização e as multiplicações em ponto-flutuante são transformadas em adições [15, 31, 32]. Diante disso, as equações apresentadas anteriormente, são modificadas para as descritas a seguir:

0. Pré-definição:

$$\begin{aligned} \pi'_i &= log(\pi_i) \\ b'_i(o_t) &= log(b_i(o_t)) \\ a'_{ij} &= log(a_i j) \\ Para \quad 1 \leq t \leq T & \qquad 1 \leq i \leq N \quad 1 \leq j \leq N \end{aligned} \tag{5.10}$$

1. Inicialização:

$$\begin{aligned} \delta_1(i)' = \pi'_i + b'_i(O_1) \quad &, 1 \leq i \leq N, \\ \psi_1(i) &= 0. \end{aligned} \tag{5.11}$$

2. Recursividade:

$$\begin{aligned} \delta_t(j)' &= \max_{1 \leq i \leq N} [\delta_{t-1}(i)' + a'_{ij}] + b_j(O_t)', \\ \psi_t(j) &= \underset{1 \leq i \leq N}{argmax} [\delta_{t-1}(i)' + a'_{ij}], \quad 2 \leq t \leq T, \quad 1 \leq j \leq N. \end{aligned} \tag{5.12}$$

3. Término:

$$\begin{aligned} P^* &= \max_{1 \leq i \leq N} [\delta_T(i)'], \\ q^*_T &= \underset{1 \leq i \leq N}{argmax} [\delta_T(i)']. \end{aligned} \tag{5.13}$$

4. Sequência de estados ótima

$$q^*_t = \psi_{t+1}(q^*_{t+1}) \quad , t = T-1, T-2, \ldots, 1. \tag{5.14}$$

5.9 Considerações Gerais

Aplicações de reconhecimento de fala utilizando métodos estatísticos exigem uma quantidade de dados de treinamento representativa, que permita a correta modelagem dos padrões a serem reconhecidos. Além disso, é necessária a investigação dos parâmetros mais adequados para cada idioma.

Neste capítulo, foram apresentados os resultados dos testes realizados a fim de identificar a melhor configuração para reconhecimento de fala contínua para o português brasileiro. Nos testes foram avaliados número de estados do HMM, número de coeficientes mel-cepstrais e número de filtros para obtenção dos coeficientes. Foram realizados testes com um número reduzido de coeficientes, de modo a avaliar o impacto dessa redução na taxa de reconheci-

mento.

Também foram apresentadas simplificações matemáticas para as etapas de cálculo das gaussianas e algortimo de Viterbi, visando suas implementações em um sistema embarcado de baixo custo.

Os resultados apresentados neste capítulo, permitiram identificar as melhores configurações, como também simplificações e adaptações que foram utilizadas na modelagem do sistema de reconhecimento de fala deste trabalho.

Capítulo 6

Modelagem da Arquitetura do Sistema de Reconhecimento

Como foi descrito anteriormente, o processo de reconhecimento de fala com HMM contínuo pode ser divido em basicamente quatro grandes etapas como ilustra a Figura 6.1.

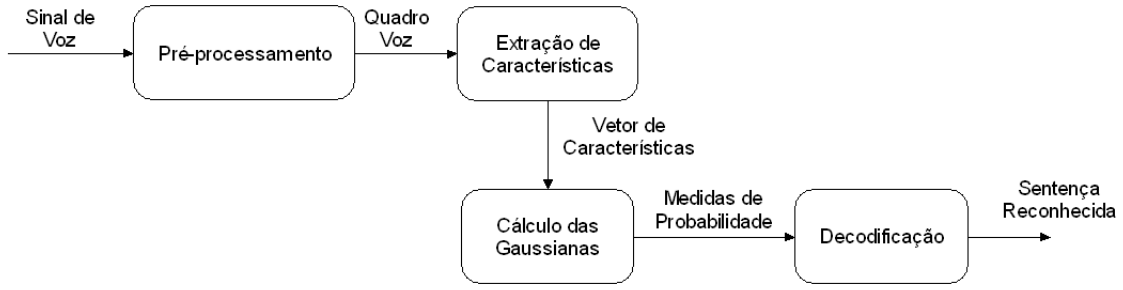


Figura 6.1: *Etapas de um sistema para reconhecimento de fala contínua com HMM contínuo.*

A etapa de pré-processamento é responsável por preparar e extrair do sinal de voz os dados relevantes e menosprezar a informação redundante, com a finalidade de repassar a informação de interesse para a etapa seguinte. Na etapa de extração de características são obtidos elementos que possibilitam a geração do padrão de teste que será comparado com os padrões de referência, conhecidos pelo sistema, e que foram criados durante a etapa de treinamento. Na etapa de cálculo das gaussianas, é calculada a probabilidade que o padrão de teste tenha sido gerado por cada um dos padrões de referência. Por fim, na etapa de Decodificação, são utilizadas as probabilidades obtidas na etapa anterior, juntamente com o Modelo Linguístico (ML) para decidir se o padrão de teste é conhecido pelo sistema e caso seja, a qual padrão de referência correspode.

Nas próximas seções, será apresentada a arquitetura de *hardware* projetada para cada uma das etapas citadas.

6.1 Pré-processamento

A arquitetura modelada para esta etapa, cujo diagrama é apresentado na Figura 6.2, tem como base o projeto descrito por Cipriano em [15], e que foi adaptado, validado e prototipado em [37].

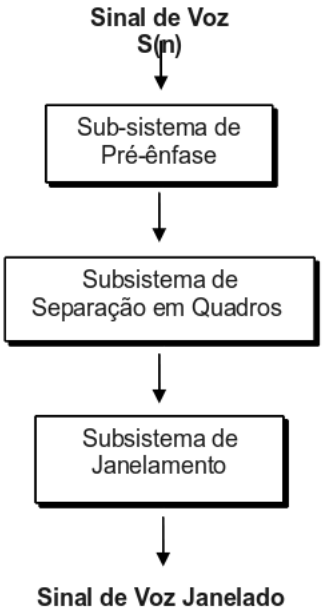


Figura 6.2: *Diagrama do Sub-sistema pré-processamento.*

O projeto apresenta adequações dos modelos necessários ao processo de reconhecimento de voz para implementação em hardware. Para isso, é feita a utilização de ponto-fixo nas operações com número fracionários, simplificações nas funções matemáticas, substituição de operações de divisão e multiplicação por deslocamentos, paralelismo, dentre outras. Na seção 6.1.1, é apresentado o subsistema de Pré-ênfase e na seção 6.1.2, o subsistema de Divisão em quadros e janelamento.

6.1.1 Pré-ênfase

A função da pré-ênfase é eliminar uma tendência espectral de aproximadamente -6dB/oitava na fala irradiada dos lábios. Essa distorção espectral não traz informação adicional e pode ser eliminada por meio da aplicação de um filtro, de resposta aproximadamente +6dB/oitava, que proporciona um nivelamento no espectro. A descrição matemática da pré-ênfase é dada por [1, 48]:

$$s_p(n) = s(n) - a_p.s(n-1), \tag{6.1}$$

sendo:

$s_p(n)$ – amostra pré-enfatizada;
$s(n)$ – amostra original;
a_p – fator de pré-ênfase, $0,9 \leq a_p \leq 1$

No intuito de simplificar a implementação do filtro de pré-ênfase, na Equação 6.1, o fator a_p é substituído pelo valor 15/16, correspondente a $0,9375$:

$$\begin{aligned} s_p(n) &= s(n) - \frac{15}{16}.s(n-1), \\ s_p(n) &= s(n) - s(n-1) + \frac{s(n-1)}{16}. \end{aligned} \tag{6.2}$$

A modificação efetuada na Equação 6.1 implica na substituição de uma multiplicação por um número fracionário, por uma divisão por um número do tipo 2^m, sendo m inteiro, e que pode ser realizada a partir de um simples deslocamento de m bits, nesse caso $m = 4$.

A arquitetura utilizada para realização da pré-ênfase (Figura 6.3) consiste de um somador, um subtrator, um acumulador, dois registradores de deslocamento e um circuito de divisão. A pré-ênfase é realizada em um único pulso de *clock*, sendo também necessários dois pulsos adicionais para a inicialização dos registradores.

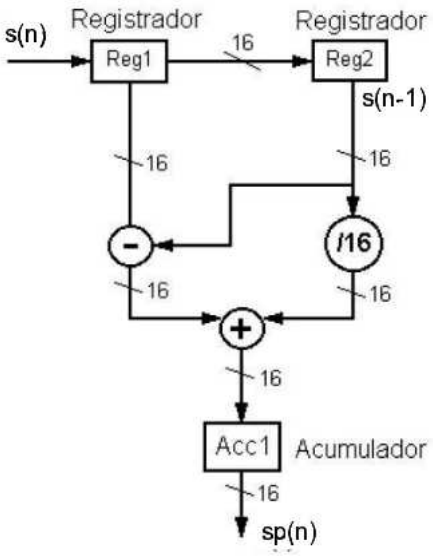


Figura 6.3: *Arquitetura da pré-ênfase.*

Na etapa de inicialização, a primeira amostra do sinal de voz digitalizado, $s(1)$, é armazenada no registrador Reg1. Na etapa seguinte, é lido um novo dado, a amostra $s(2)$. Este novo dado é armazenado em Reg1 e o valor anterior de Reg1 é transferido para Reg2. As outras operações são realizadas em paralelo e o resultado final é armazenado no acumulador Acc1. Este procedimento é repetido até o processamento total das amostras do sinal de voz. A divisão por 16 da Equação 6.2, é obtida fazendo-se um deslocamento de quatro bits para à direita, repetindo quatro vezes o bit mais significativo, à esquerda (Figura 6.4). Essa

repetição é necessária para que seja mantido o sinal do valor correspondente da amostra de voz. Tal solução simples permite realizar a divisão em um único pulso de clock.

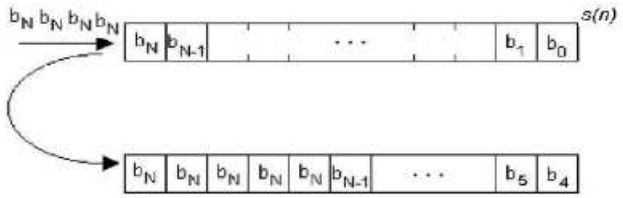


Figura 6.4: *Divisão na etapa de pré-ênfase.*

6.1.2 Divisão em quadros e Janelamento

Um sinal de voz possui características estatísticas que variam fortemente com o tempo, podendo ser considerado estacionário para curtos intervalos de tempo. A separação em quadros consiste em particionar o sinal de voz em segmentos, selecionados por janelas ou "frames" de duração perfeitamente definida . O tamanho desses segmentos é escolhido dentro dos limites de estacionariedade do sinal (duração média de 16 a 32 ms) [17, 40]. No sistema proposto o algoritmo de janelamento utilizado foi o de Hamming, que minimiza as descontinuidades no início e final de cada quadro, e cuja equação é dada por 6.3, sendo N_A o número de amostras de um segmento de análise.

$$J(n) = \begin{cases} 0,54 - 0,46cos[2\pi n/(N_A - 1)] & ,0 \leq n \leq N_A - 1 \\ 0 & ,\text{caso contrário.} \end{cases} \tag{6.3}$$

O janelamento é realizado em conjunto com a separação dos quadros, que contêm 252 (N_A) amostras cada. Cada amostra do quadro é multiplicada pelo valor correspondente da janela, cujo tamanho também é igual a 252. A utilização de 252 amostras corresponde a um quadro com aproximadamente 23 ms, valor este que atende ao intervalo especificado por [17], o que garante a estacionariedade da parte analisada do sinal. Além disso, o número 252 é divisível por 3 e por 9, o que facilita a implementação do algoritmo de janelamento com superposição utilizado (Figura 6.5), uma vez que, cada quadro é formado por 3 blocos, sendo que cada bloco é escrito em 3 etapas, daí a facilidade oferecida por 252 ser divisível por 9. O algoritmo de escrita das amostras será explicado mais adiante.

Para formar os quadros, as amostras depois de serem processadas pela função de pré-ênfase são inicialmente segmentadas em blocos, em que cada bloco contém 84 amostras, não existindo superposição entre blocos, mas apenas entre quadros. Cada quadro, Q_i, será composto por três blocos sucessivos, totalizando 252 amostras. A divisão dos quadros em 3 blocos de 84 amostras, ou seja, 1/3 do tamanho do quadro, possibilita uma superposição de aproximadamente 66%, como pode ser observado na Figura 6.5. À medida que cada quadro

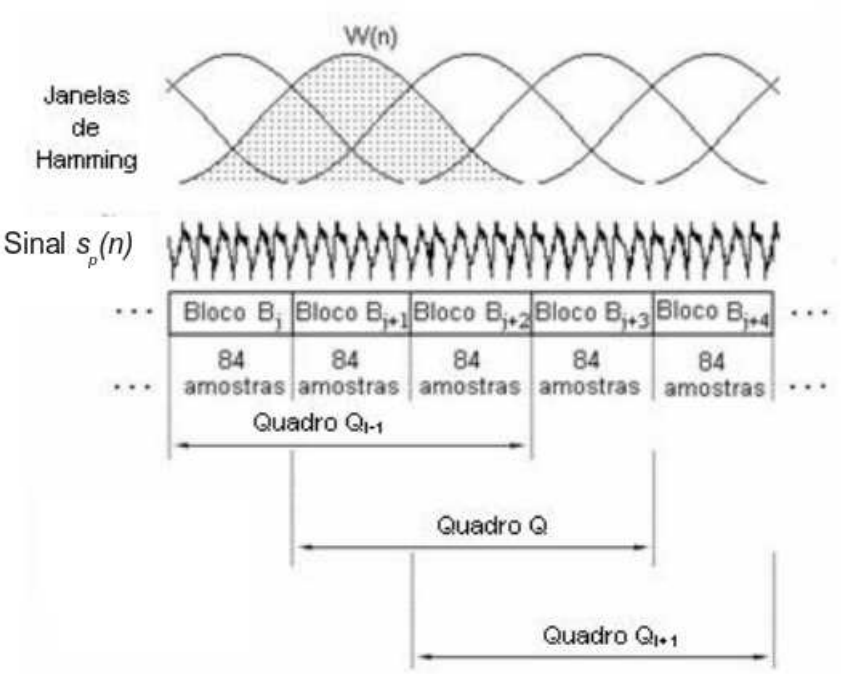


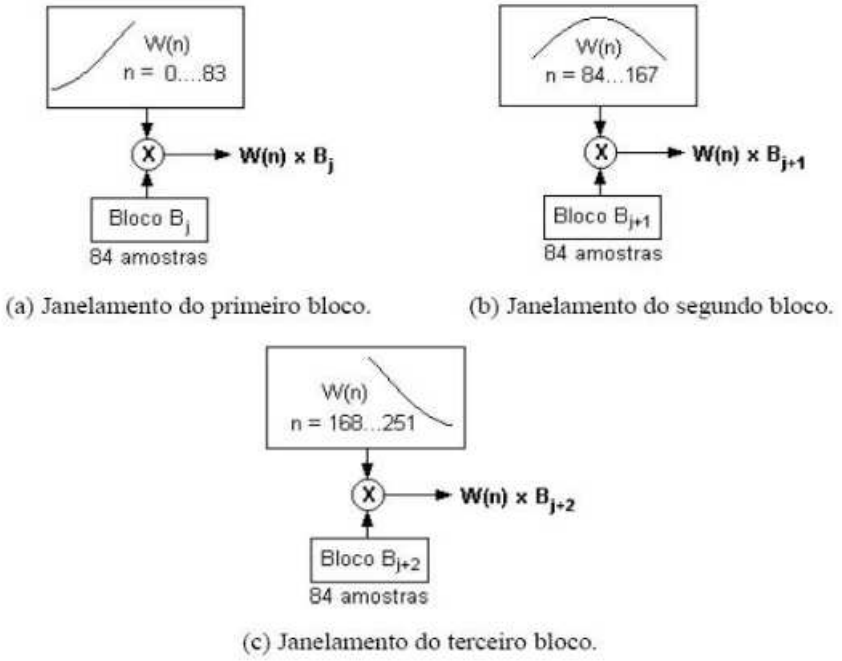
Figura 6.5: *Divisão em quadros e janelamento.*

vai sendo formado, suas amostras vão sendo multiplicadas pelo valor da janela de *Hamming* correspondente, como é ilustrado na Figura 6.6.

Para a implementação em hardware do algoritmo de divisão em quadros, utiliza-se uma memória RAM RW de dupla porta, que permite a leitura e a escrita de forma paralela. Esta memória é dividida em três segmentos *M(0)*, *M(1)* e *M(2)*, de 84 amostras cada um. Em cada segmento de memória serão armazenadas as amostras dos blocos que formarão um quadro. O resultado da multiplicação de cada amostra do quadro final pelo valor correspondente da janela de *Hamming* é armazenado em um registrador.

Os quadros são formados pelo agrupamento dos segmentos da memória. Para permitir o trabalho em paralelo das operações de leitura e escrita na memória, a sequência em que os segmentos de memória são agrupados ocorre ciclicamente, como mostrado na Figura 6.7.

As amostras do primeiro bloco (o bloco B_0) não precisam ser armazenadas na memória, pois serão processadas diretamente, multiplicando-as pela porção da janela de *Hamming* correspondente. As amostras do segundo bloco (o bloco B_1) são armazenadas no segmento de memória *M(0)* e as amostras do terceiro bloco (o bloco B_2) são armazenadas no segmento de memória *M(1)*. Assim, o primeiro quadro Q_0 será formado pelas amostras do primeiro bloco (que não precisou ser armazenado) e as amostras armazenadas nos segmentos de memória *M(0)* e *M(1)*. Em um tempo posterior, as amostras do quarto bloco (o bloco B_3) são armazenadas no segmento de memória *M(2)*. Neste momento não é necessário alterar o conteúdo dos segmentos *M(0)* e *M(1)*. Assim, tem-se o segundo quadro, Q_1, que é formado pelas amostras armazenadas nos segmentos de memória *M(0)*, *M(1)* e *M(2)*. Continuando com o processamento, as amostras do quinto bloco (o bloco B_4) são armazenadas no segmento



(a) Janelamento do primeiro bloco.

(b) Janelamento do segundo bloco.

(c) Janelamento do terceiro bloco.

Figura 6.6: *Multiplicação dos blocos pela janela de Hamming.*

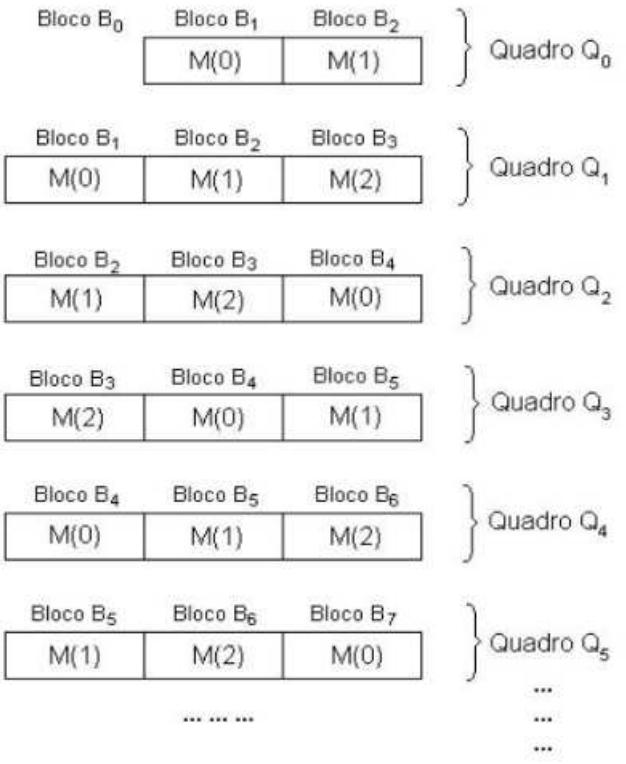


Figura 6.7: *Sequência de blocos e seu armazenamento nos segmentos da memória.*

de memória *M(0)*. O conteúdo anterior deste segmento (o bloco B_1), não será mais necessário para o processamento posterior. Neste momento não é necessário alterar os segmentos *M(1)* e *M(2)*, que contêm os blocos B_2 e B_3. Assim, tem-se o terceiro quadro, Q_2, o qual é formado pelas amostras armazenadas nos segmentos de memória *M(1)*, *M(2)* e *M(0)*, gerando uma nova sequência de segmentos. Na continuação, as amostras do sexto bloco (o bloco B_5) são armazenadas no segmento *M(1)*. O conteúdo anterior deste segmento (o bloco B_2), não será mais utilizado no processamento posterior. No entanto, o conteúdo dos segmentos *M(2)* e *M(0)* não é alterado (blocos B_3 e B_4), neste momento. Assim, tem-se o quarto quadro (o quadro Q_3), formado pelas amostras armazenadas nos segmentos de memória *M(2)*, *M(0)* e *M(1)*. O processamento dos demais blocos é realizado segundo o algoritmo explicado anteriormente, o que implica nas sequências de escrita e leitura apresentadas na Figura 6.8.

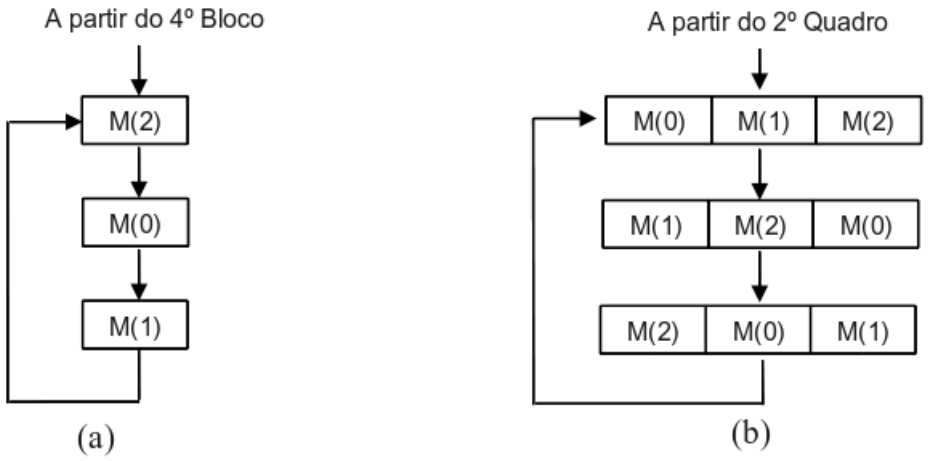


Figura 6.8: *Sequências de Escrita dos Blocos (a) e Leitura dos Quadros (b).*

A escrita de amostras nos segmentos da memória é realizada paralelamente com a operação de leitura dos segmentos. Isto significa que, enquanto um segmento da memória está sendo escrito, outro segmento da memória está sendo lido e multiplicado pela janela de *Hamming*. Para tanto, foi necessária uma velocidade da operação de leitura da memória igual ao triplo da velocidade da operação de escrita. Isto significa que, enquanto são lidas 84 amostras que estão armazenadas em um segmento da memória, são recebidas do sinal de voz e armazenadas em um outro segmento apenas 28 amostras. Para ilustrar o funcionamento desse algoritmo (Figura 6.9), considera-se o quadro Qi formado pelos blocos B_j, B_{j+1} e B_{j+2}, armazenados nos segmentos *M(0)*, *M(1)* e *M(2)* respectivamente. O quadro anterior Q_{i-1} foi formado pelos blocos B_{j-1}, B_j e B_{j+1}, que ocupavam os segmentos de memória *M(2)*, *M(0)* e *M(1)*, respectivamente.

Quando o último bloco de $Q_{i-1}(B_{j+1})$, armazenado em *M(1)* está sendo processado, as primeiras 28 amostras de B_{j+2} de Q_i já estão sendo armazenadas em *M(2)*, em seguida, ao iniciar o processamento de Q_i, com a leitura de B_j, armazenado em *M(0)*, paralelamente a segunda parte de B_{j+2} é escrita em *M(2)*. Novamente é lido o conteúdo de *M(1)*, que ainda

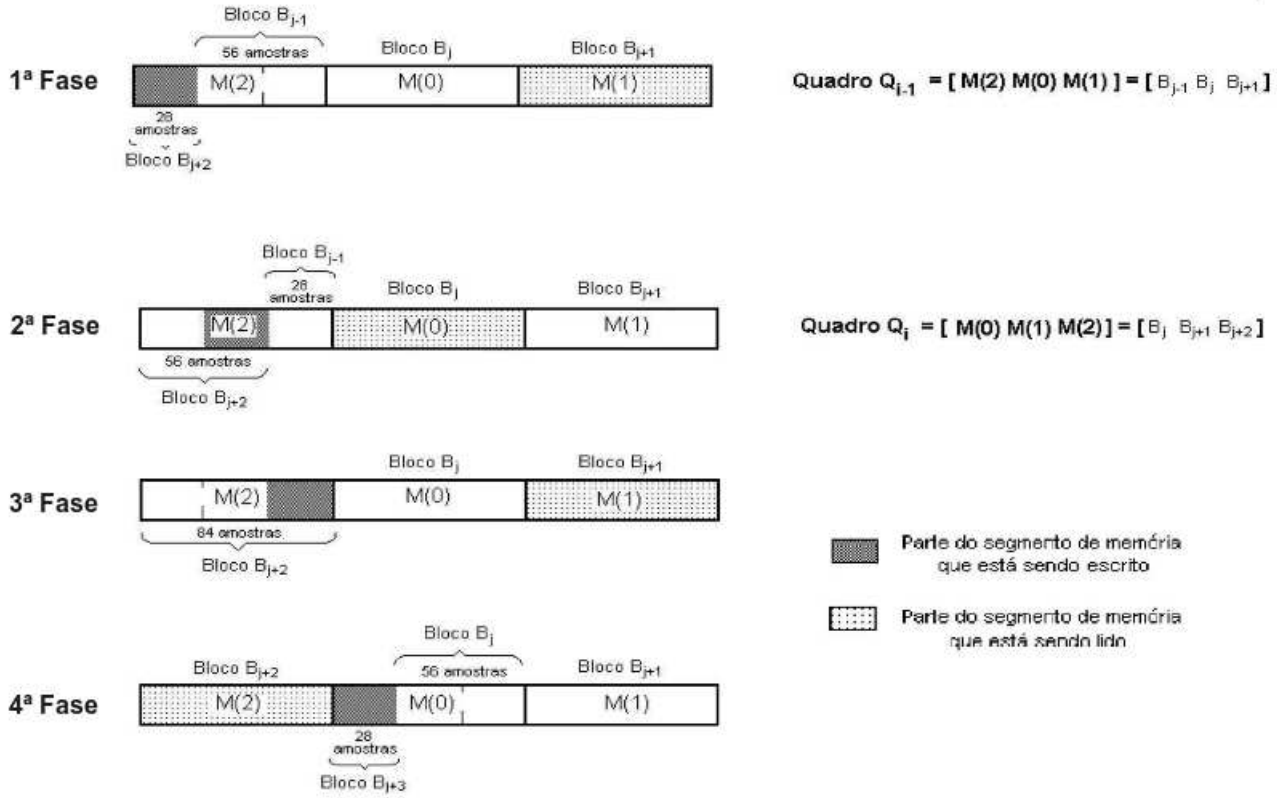


Figura 6.9: *Operações de escrita e leitura nos segmentos de memória.*

contém B_{j+1} (segunda parte de Q_i), enquanto finalmente é escrita a última parte de B_{j+2} em *M(2)*, referente a último bloco de Q_i. O processo continua com a escrita da primeira parte do bloco seguinte (B_{j+3}) em *M(0)*, que será o último segmento lido do próximo quadro Q_{i+1}.

A implementação da janela de Hamming foi realizada com uma memória ROM que armazena os primeiros 126 valores da janela $W(n)(n = 0, ..., 125)$. A partir destes valores são obtidos os 126 valores restantes de $W(n)(n = 126, ..., 251)$, utilizando a propriedade de simetria da janela (Figura 6.10).

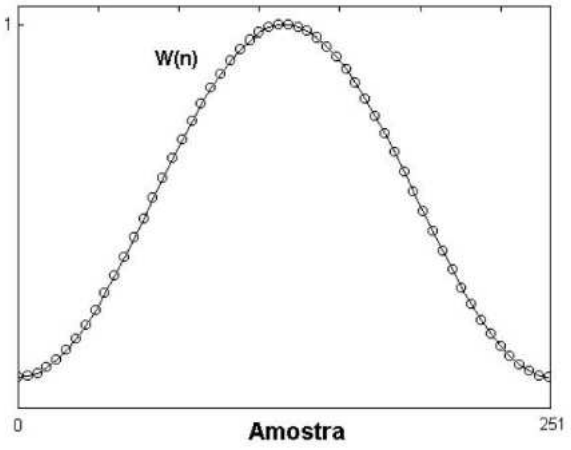


Figura 6.10: *Janela de Hamming.*

O registrador de endereçamento da memória que contêm os valores da janela de Hamming é implementado com um contador binário de 0...125, que opera nos modos crescente/decrescente. Na Figura 6.11, é apresentada parcialmente a arquitetura do módulo da função de divisão de quadros e janelamento. Os valores da janela de *Hamming* encontram-se armazenados na memória *rom_ memory*, de 12 bits, de onde são lidos e multiplicados pelos valores do quadro do sinal de voz pré-enfatizado.

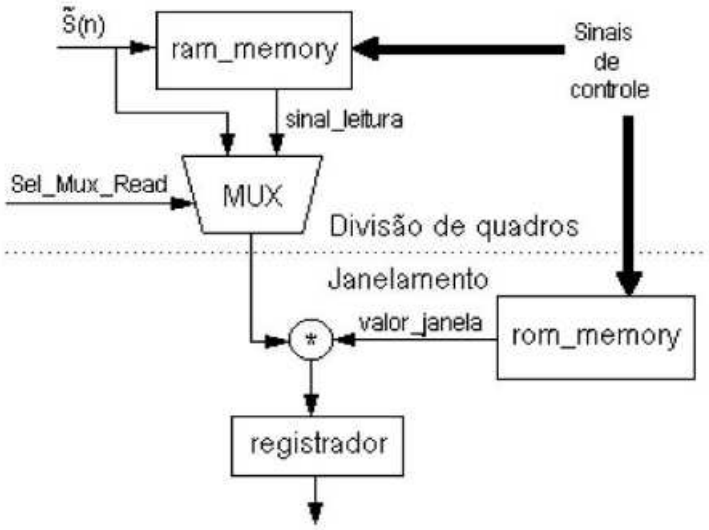


Figura 6.11: *Arquitetura do módulo da função de Divisão em Quadros e Janelamento.*

Os valores de um quadro do sinal de voz pré-enfatizado se encontram armazenados na memória *ram_ memory*. A memória *ram_ memory* é uma memória de duplo acesso síncrona, de 16 bits, que permite a leitura e a escrita de valores em paralelo. Essa memória foi dividida em três segmentos, para a implementação do algoritmo de separação em quadros explicado anteriormente. Foi incluído um multiplexador que seleciona, a partir do sinal *Sel_ Mux_ Read*, entre realizar a leitura de um quadro Q_i a partir da memória RAM RW ou a partir do sinal pré-enfatizado. Isso permite o processamento do primeiro bloco de 84 amostras que é multiplicado diretamente pela janela, sem necessidade de lê-lo da memória RAM RW. Para multiplicar a janela de *Hamming* por cada quadro formado, a partir do sinal de voz, utilizou-se um multiplicador de 12 bits x 16 bits e o resultado de 28 bits foi reduzido para 16 bits e armazenado em um registrador de saída. Na Figura 6.12, é apresentada a descrição de como é realizado o endereçamento das memórias utilizadas. A leitura da memória ROM é realizada à medida que vai sendo calculado o janelamento das amostras e é controlada pelo componente *up_ down counter* com duas frequências de *clock* diferentes f_{ck2} e f_{ck1}, sendo $f_{ck2} = 3f_{ck2}$. A frequência f_{ck1} é a frequência com que os dados do sinal de voz são recebidos da pré-ênfase. Como foi mencionado anteriormente, o primeiro bloco não é lido da memória, e sim processado diretamente do sinal de voz pré-enfatizado, o que torna necessária a leitura dos valores da janela de *Hamming*, nesse instante, com a frequência f_{ck1}. No entanto, a partir do segundo bloco todos os valores são lidos da memória RAM RW. Como a escrita na memória RAM é realizada conforme as amostras do sinal de voz pré-enfatizado são recebidas, ou

seja, com frequência f_{ck1} e devem ser lidos três blocos da memória RAM, enquanto é escrito apenas um, a frequência de leitura da RAM deve ser três vezes a frequência de escrita, ou seja, f_{ck2}.

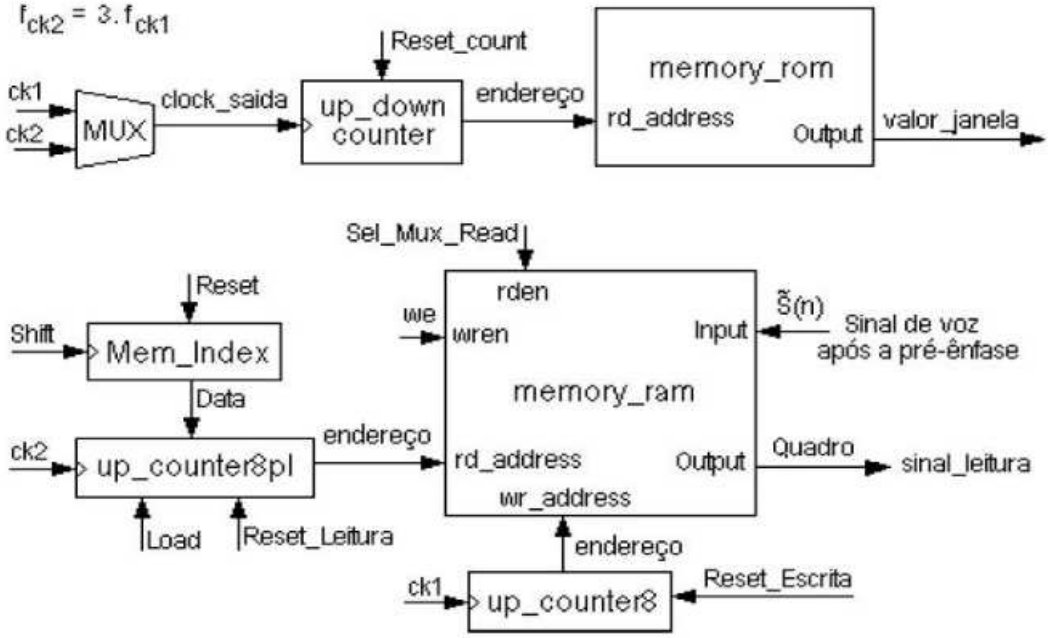


Figura 6.12: *Endereçamento das memórias RAM RW e ROM nas funções de divisão em quadros e janelamento.*

A memória RAM RW utiliza o contador *up_ counter8*, de 8 bits, para o cálculo dos endereços de escrita. Na memória são armazenadas 252 amostras, portanto, uma vez que o contador atinge o valor 251 ele é reinicializado e a contagem recomeça. O cálculo dos endereços de leitura da memória RAM RW é feito por meio de um contador *up_ counter8pl*, de 8 bits, com uma entrada *Load* para carregar o valor a partir do qual o contador inicia a contagem. Isto permite selecionar o endereço da RAM RW a partir do qual as amostras serão lidas. Este endereço pode ser 0, 84 ou 168, que são os endereços onde começam os três segmentos em que a RAM RW foi dividida. Este valor é fornecido pelo circuito *Mem_ Index* apresentado na Figura 6.13. O circuito *Mem_ index* foi implementado com três registradores em cascata e um contador *Counter7*, de 7 bits, que faz a contagem de 0, ..., 83, contando o número de valores lidos de um segmento de memória (até 84 valores). Em cada registrador é armazenado um dos endereços que cada segmento de memória inicia e o sinal de controle *Shift* servirá como *clock* indicando a circulação dos valores entre cada registrador. Na saída, Data, ter-se-á o endereço inicial para ser carregado no contador *up_ counter8pl*.

6.2 Extração de Características

Depois da aplicação da janela de *Hamming*, é realizado o cálculo dos parâmetros mel-cepstrais de cada quadro. Neste trabalho, a arquitetura foi projetada para calcular 9 pa-

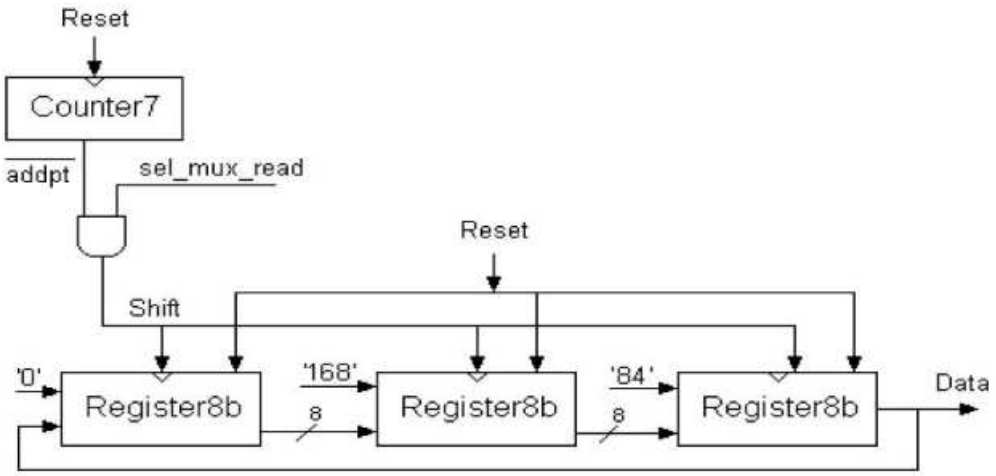


Figura 6.13: *Componente Mem_ Index.*

râmetros mel-cepstrais, juntamente com suas primeira e segunda derivadas. Na Figura 6.14 é ilustrado o processo de obtenção dos parâmetros mel-cepstrais.

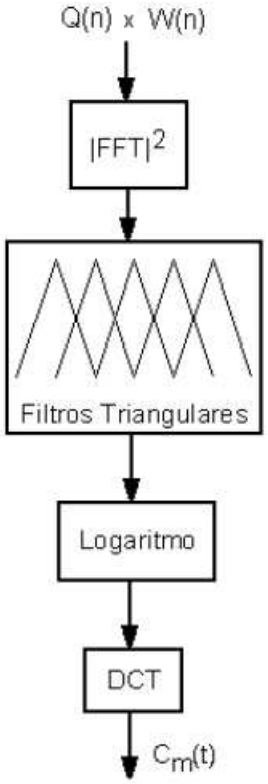


Figura 6.14: *Extração dos parâmetros mel-cepstrais.*

Os quadros do sinal de voz, após terem sido multiplicados pela janela de *Hamming*, são processados da seguinte maneira:

1. Calcula-se o espectro de energia por meio da FFT;

2. Por meio de um banco de filtros triangulares, calcula-se a energia de cada canal;

3. Calcula-se o logaritmo do valor da energia de cada canal;

4. Calcula-se a transformada cosseno.

Nas seções seguintes, é apresentada a arquitetura de cada uma dessas etapas.

6.2.1 Função FFT

A função FFT implementa a seguinte equação:

$$X(k) = \sum_{n=0}^{N_{w-1}} S_Q(n).e^{\frac{-j2\pi kn}{Nw}}, 0 \leq k \leq N_{w-1}, \tag{6.4}$$

sendo $S_Q(n)$ é o sinal de voz proveniente da separação em quadros após ter sido processado pela janela de *Hamming*. $N_w = 252$ corresponde ao número de amostras dentro de um quadro. O espectro de energia é dado por:

$$F_k = |X(k)|^2, 0 \leq k \leq \frac{N_w}{2} - 1. \tag{6.5}$$

A função FFT implementa a Equação 6.4 iterativamente, a partir das seguintes equações:

$$\begin{aligned} X_l(k) &= X_{l-1}(k) + W^p.X_{l-1}(k + \frac{N}{2^l}), \\ X_l(k + \frac{N}{2^l}) &= X_{l-1}(k) - W^p.X_{l-1}(k + \frac{N}{2^l}), \end{aligned} \tag{6.6}$$

sendo $X_l(k)$ é o valor de $X(k)$ na iteração l. O número de iterações necessário para obter os valores finais de $X(k)$ é $l = log_2 N$. Foram utilizadas 256 amostras, pois os 252 valores provenientes da divisão em quadros foram completados com zeros, para obter um número de amostras de entrada do tipo 2^n, sendo n inteiro. O fator W_p é definido como:

$$W^p = e^{\frac{-j2\pi p}{Nw}}, p = n.k \tag{6.7}$$

Na Figura 6.15, é apresentada a estrutura de operações necessárias para a implementação da FFT. Duas memórias ROM *Twiddle_Re_Mem* e *Twiddle_Im_Mem* de 128 palavras de 16 bits são utilizadas para armazenar a parte real e a parte imaginária do fator W_p, da equação 6.7. Quatro memórias *RAM Re0, Im0, Re1 e Im1*, de 128 palavras de 16 bits cada uma, são utilizadas para armazenar os valores intermediários de cada iteração. Estas memórias devem ter dupla porta de acesso.

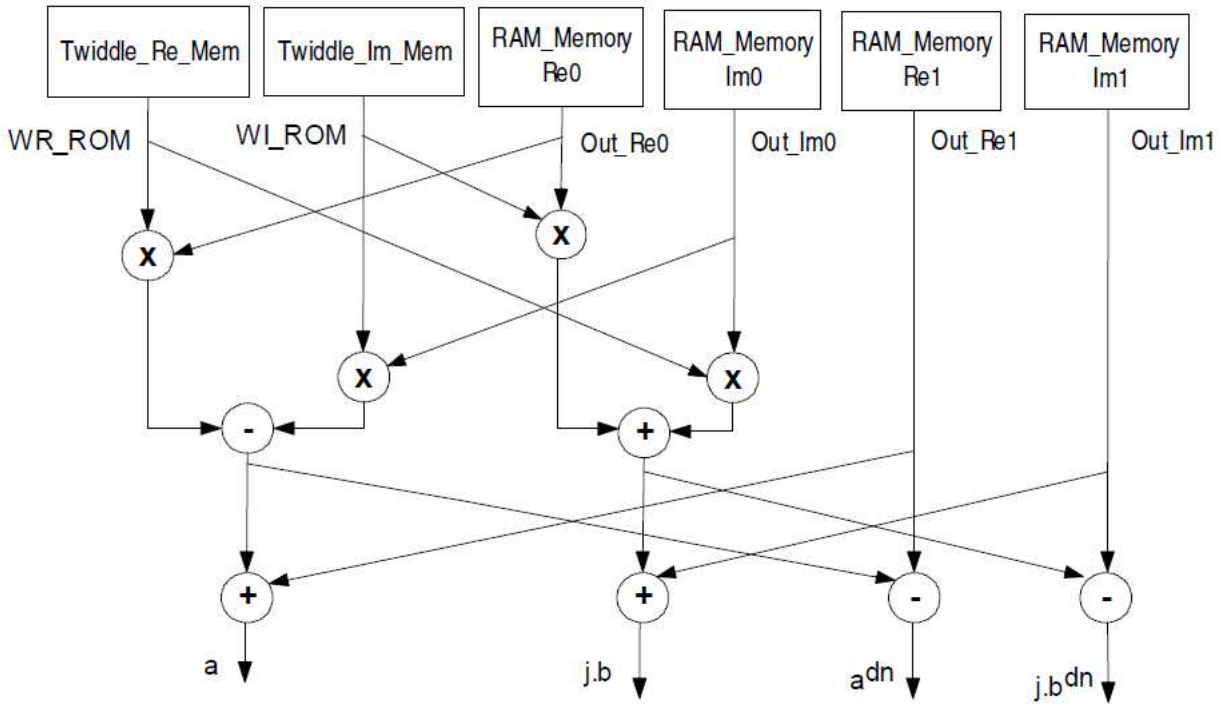


Figura 6.15: *Operações cruzadas para a FFT (butterfly).*

As saídas a cada iteração foram separadas, para facilitar o processamento em *hardware*, em suas partes real e imaginária da seguinte maneira:

$$\begin{aligned} a &= ReX_l(k), \\ b &= ImX_l(k), \\ a^{dn} &= ReX_l(k + \frac{N}{2^l}), \\ b^{dn} &= ImX_l(k + \frac{N}{2^l}). \end{aligned} \tag{6.8}$$

Na Figura 6.16, é ilustrada a entrada de dados nas memórias RAM RW, que armazenam os valores intermediários em cada iteração.

Os valores de $X(k)$ a cada iteração são novamente introduzidos nas memórias RAM RW para o cálculo das iterações seguintes.

6.2.2 Função de Filtros Triangulares

O espectro do sinal de voz é processado por um banco de filtros triangulares, similar ao apresentado na Figura 2.9, no Capítulo 2.

Com base nos experimentos realizados, cujos resultados foram apresentados no Capítulo 5, e também em outros trabalhos relacionados [147, 148, 149, 150, 151], foram utilizados 36 filtros triangulares para cobrir a faixa de frequências do sinal de voz. Uma vez que a frequência de amostragem utilizada foi de $f_s = 11025Hz$, respeitando o critério de Nyquist,

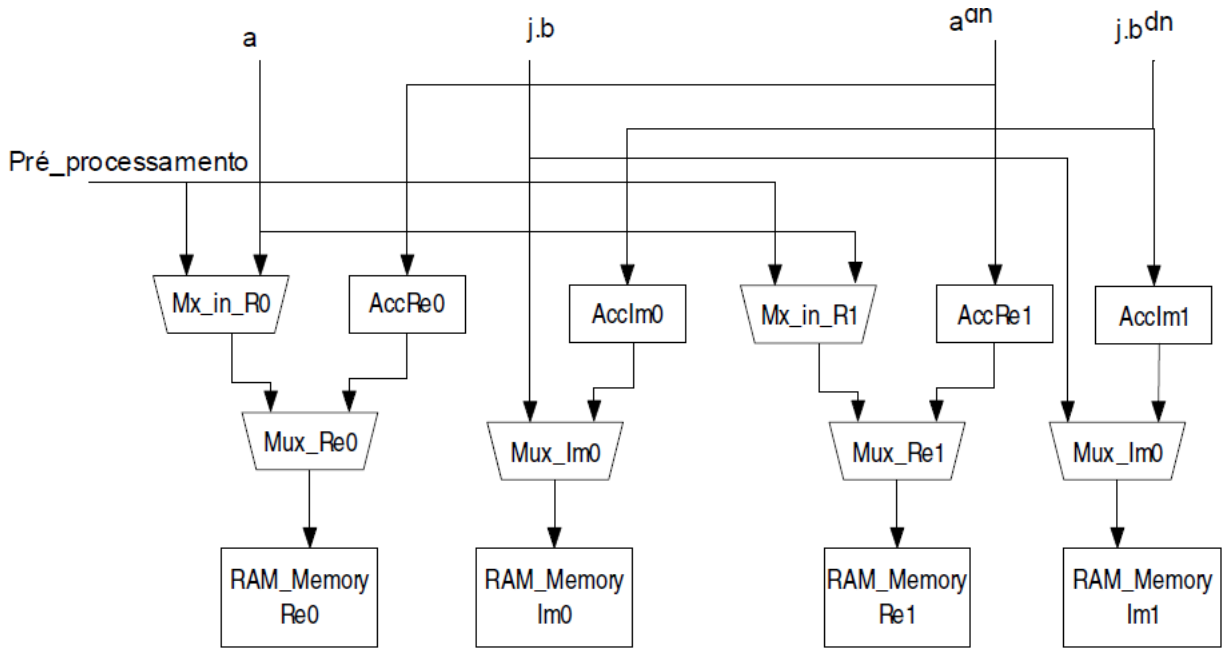


Figura 6.16: *Entrada de dados nas memorias RAM RW.*

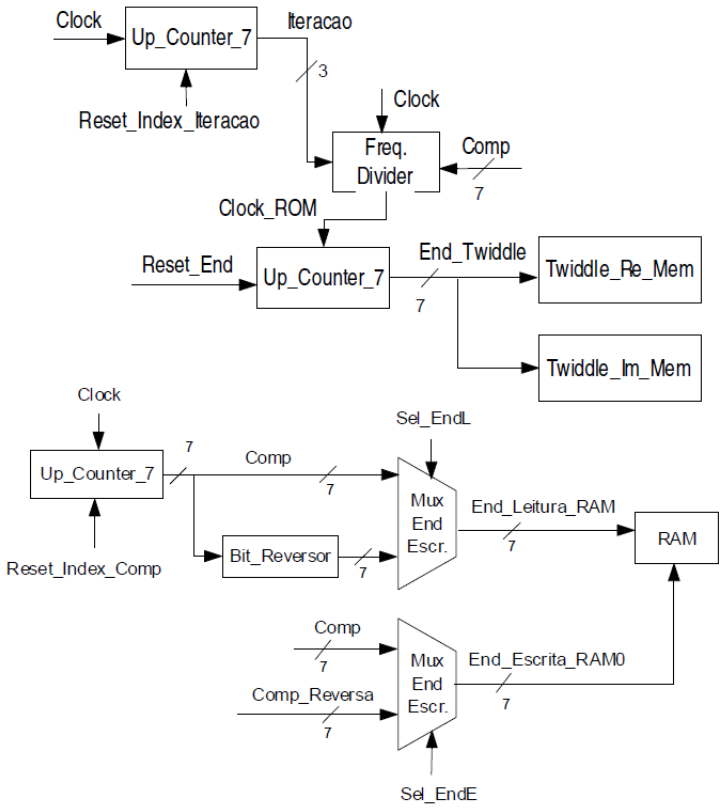


Figura 6.17: *Endereçamento das Memórias.*

a maior frequência não deve ultrapassar a metade desse valor. Na Tabela 6.1 são apresentadas as frequências do banco de filtros implementado.

Índice	Menor Frequência (Hz)	Frequência Central (Hz)	Maior Frequência (Hz)	Distância do Centro	Largura de Banda
1	133.33	200.00	266.67	66.67	133.34
2	200.00	266.67	333.34	66.67	133.34
3	266.67	333.34	400.01	66.67	133.34
4	333.34	400.01	466.68	66.67	133.34
5	400.01	466.68	533.35	66.67	133.34
6	466.68	533.35	600.02	66.67	133.34
7	533.35	600.02	666.69	66.67	133.34
8	600.02	666.69	733.36	66.67	133.34
9	666.69	733.36	800.03	66.67	133.34
10	733.36	800.03	866.70	66.67	133.34
11	800.03	866.70	933.37	66.67	133.34
12	866.70	933.37	1000.04	66.67	133.34
13	933.37	1000.04	1066.71	66.67	133.34
14	1000.04	1071.38	1142.71	71.34	142.67
15	1071.38	1147.71	1224.04	76.33	152.66
16	1147.71	1229.38	1311.05	81.67	163.35
17	1229.38	1316.77	1404.16	87.39	174.78
18	1316.77	1410.28	1503.79	93.51	187.02
19	1410.28	1510.33	1610.39	100.05	200.11
20	1510.33	1617.39	1724.45	107.06	214.11
21	1617.39	1731.94	1846.49	114.55	229.10
22	1731.94	1854.51	1977.08	122.57	245.14
23	1854.51	1985.66	2116.81	131.15	262.30
24	1985.66	2125.99	2266.32	140.33	280.66
25	2125.99	2276.15	2426.30	150.15	300.31
26	2276.15	2436.81	2597.48	160.66	321.33
27	2436.81	2608.72	2780.63	171.91	343.82
28	2608.72	2792.67	2976.61	183.94	367.89
29	2792.67	2989.49	3186.31	196.82	393.64
30	2989.49	3200.09	3410.68	210.60	421.20
31	3200.09	3425.43	3650.77	225.34	450.68
32	3425.43	3666.54	3907.65	241.11	482.23
33	3666.54	3924.53	4182.52	257.99	515.98
34	3924.53	4200.58	4476.63	276.05	552.10
35	4200.58	4495.96	4791.33	295.37	590.75
36	4495.96	4812.01	5128.06	316.05	632.10

Tabela 6.1: *Banco de Filtros Triangulares.*

Como apenas metade do espectro de frequência é considerada, a entrada da função dos filtros triangulares consiste de um vetor de 128 valores, o que corresponde a metade do vetor gerado pela função FFT.

Como descrito no Capítulo 2, e repetido aqui por conveniência, nesta etapa é calculada a energia de saída $P(i)$ em cada filtro i, expressa por:

$$P(i) = \sum_{k=0}^{N/2} |S(k,m)|^2 H_i(k\frac{2\pi}{N}), \tag{6.9}$$

com $i = 1, 2, ..., N_f$. N é o número de pontos da Transformada Discreta de Fourier, N_f é o número de filtros triangulares, que neste caso é igual 36. $|S(k, m)|$ é o módulo da amplitude na frequência do k-ésimo ponto da m-ésima janela (quadro). $H_i(w)$ é a função resposta em frequência do i-ésimo filtro triangular.

Como apenas 128 valores do espectro de frequência passam pelo banco de filtros então, também são necessários apenas 128 valores de cada filtro. No entanto, para cada filtro, muitos dos seus valores são iguais a zero. Sendo assim, para evitar o cálculo dos valores dos filtros em tempo de execução, os valores diferentes de zero são previamente calculados em *software* e armazenados em uma memória ROM, *Valor_Filtro_Rom*, de 261 palavras de 12 bits cada. São necessários 261 endereços de memória, uma vez que, para os 36 filtros utilizados com as frequências apresentadas na Tabela 6.1, existem 226 valores diferentes de zero e são utilizados outros 35 valores iguais a 0 para os demais casos. Todos os valores dos filtros podem ser verificados nas Tabelas B.1, B.2 e B.3 no Apêndice B. Em uma outra memória ROM, *Ini_Fin_Filtro_ROM*, são armazenadas as posições iniciais e finais, entre 0 e 127, dos valores diferentes de zero utilizados em cada filtro.

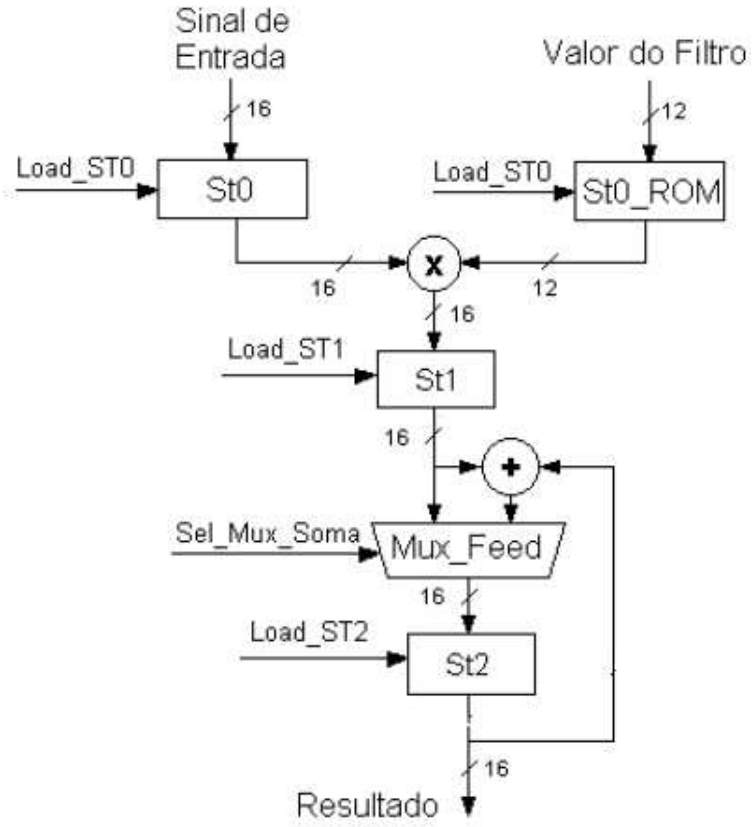


Figura 6.18: *Parte operacional da função dos filtros triangulares.*

Na Figura 6.18, é apresentada a parte operacional da função de filtros triangulares. Os registradores *St0, St0_ROM, St1* e *St2* implementam o seguinte *pipeline* de 3 estágios:

1. No primeiro estágio, cada amostra do espectro proveniente do bloco FFT é armazenada no registrador St0. Ao mesmo tempo, o ganho do filtro triangular correspondente é

lido de Valor_Filtro_ROM e armazenado em St0_ROM.

2. No segundo estágio, os valores dos registradores St0 e St0_ROM são lidos e multiplicados, e o resultado armazenado em St1.

3. No terceiro estágio, realiza-se a soma com o valor que está em *St2*. Este processo de acumulação é repetido até completar todos os valores que correspondem a um canal do banco de filtros. Uma vez concluído este processo para um canal, o multiplexador Mux_Feed reinicializa o acumulador *St2* com o primeiro valor correspondente ao próximo canal. O processo é repetido até completar todos os canais.

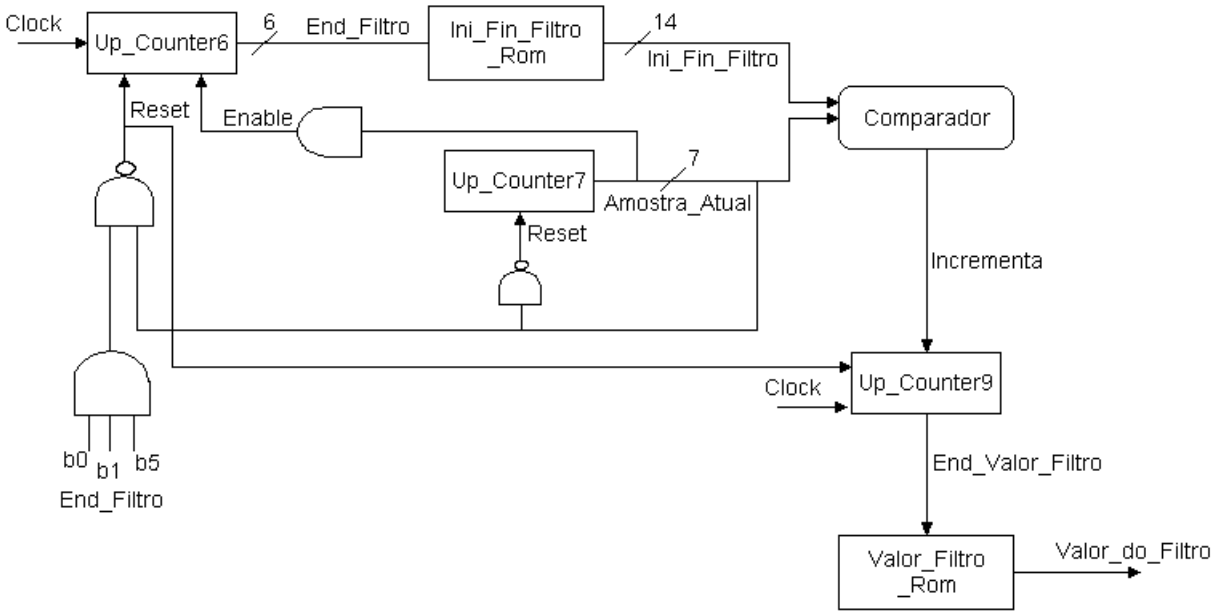


Figura 6.19: *Endereçamento das memórias ROM.*

As memórias são endereçadas por meio de contadores, conforme ilustrado na Figura 6.19. Um contador de 6 bits (Up_Counter6) é utilizado para endereçar *Ini_Fin_Filtro_ROM* que armazena as posições iniciais e finais, entre 0 e 127, dos valores utilizados em cada um dos 36 filtros. Essa memória é formada por 36 palavras de 14 bits cada uma, sendo os 7 primeiros bits para a posição inicial e os demais para a posição final. O conteúdo dessa memória é ilustrado na Tabela 6.2. Tais valores são resultantes das Tabelas B.1, B.2 e B.3 do Apêndice B, citadas anteriormente. O contador *Up_Counter7* controla o número da amostra atual que está sendo processada (0-127). O componente *Comparador* verifica se a amostra atual está entre a amostra inicial e final do filtro atual. Se isso for verdade, o contador de 9 bits, *Up_Counter9*, é incrementado (Figura 6.20). Esse contador serve para endereçar *Valor_Filtro_Rom* que armazena os valores de cada filtro, bem como o valor zero intermediário entre cada filtro. O conteúdo desta ROM pode ser verificado na Tabela 6.3. Observando o conteúdo da tabela, é possível notar que entre os valores diferentes de zero de cada filtro, existe um valor igual a zero, que vai ser utilizado até que o próximo filtro seja processado e a amostra atual esteja na faixa definida pelo próximo *Ini_Fin_Filtro*.

Endereço	Inicial	Final	Endereço	Inicial	Final
0	0	5	18	32	36
1	4	6	19	35	39
2	6	8	20	37	42
3	7	9	21	40	45
4	9	11	22	43	48
5	10	12	23	46	51
6	12	14	24	49	55
7	13	16	25	52	59
8	15	17	26	56	63
9	17	19	27	60	68
10	18	20	28	64	73
11	20	22	29	69	78
12	21	23	30	74	84
13	23	25	31	79	90
14	24	27	32	85	96
15	26	29	33	91	103
16	28	31	34	97	110
17	30	34	35	104	118

Tabela 6.2: *Conteúdo da memória Ini_Fin_Filtro_ROM.*

```
1   Divide INI_FIN_FILTRO (14 Bits):
2     INI = INI_FIN_FILTRO[7-13]
3     FIN = INI_FIN_FILTRO[0-6]
4
5   Se INI <= ATUAL <= FIN
6     Incrementa = 1
7   ELSE
8     Incrementa = 0
```

Figura 6.20: *Lógica do componente Comparador.*

End	Valor	End	Valor	End	Valor	End	Valor	End	Valor	End	Valor
0	0.2154	46	0.4347	92	**0.0000**	138	0.1046	184	0.9175	230	**0.0000**
1	0.4307	47	**0.0000**	93	0.1670	139	0.3551	185	0.7389	231	0.0675
2	0.6460	48	0.5653	94	0.5430	140	0.6056	186	0.5603	232	0.2133
3	0.8614	49	0.7887	95	0.9190	141	0.8561	187	0.3817	233	0.3591
4	0.7701	50	0.1426	96	0.7244	142	0.9004	188	0.2031	234	0.5049
5	0.1240	51	**0.0000**	97	0.3730	143	0.6662	189	0.0245	235	0.6507
6	**0.0000**	52	0.2113	98	0.0216	144	0.4321	190	**0.0000**	236	0.7965
7	0.2299	53	0.8574	99	**0.0000**	145	0.1980	191	0.0825	237	0.9423
8	0.8760	54	0.5297	100	0.2756	146	**0.0000**	192	0.2611	238	0.9177
9	0.4782	55	**0.0000**	101	0.6270	147	0.0996	193	0.4397	239	0.7814
10	**0.0000**	56	0.4703	102	0.9784	148	0.3338	194	0.6183	240	0.6452
11	0.5218	57	0.9308	103	0.6919	149	0.5679	195	0.7969	241	0.5089
12	0.8322	58	0.3666	104	0.3635	150	0.8020	196	0.9755	242	0.3726
13	0.1861	59	**0.0000**	105	0.0352	151	0.9662	197	0.8559	243	0.2364
14	**0.0000**	60	0.0692	106	**0.0000**	152	0.7474	198	0.6890	244	0.1001
15	0.1678	61	0.6334	107	0.3081	153	0.5286	199	0.5221	245	**0.0000**
16	0.8139	62	0.8154	108	0.6365	154	0.3098	200	0.3551	246	0.0823
17	0.5403	63	0.2880	109	0.9648	155	0.0910	201	0.1882	247	0.2186
18	**0.0000**	64	**0.0000**	110	0.7259	156	**0.0000**	202	0.0213	248	0.3548
19	0.4597	65	0.1846	111	0.4190	157	0.0338	203	**0.0000**	249	0.4911
20	0.8943	66	0.7120	112	0.1122	158	0.2526	204	0.1441	250	0.6274
21	0.2482	67	0.7763	113	**0.0000**	159	0.4714	205	0.3110	251	0.7636
22	**0.0000**	68	0.2836	114	0.2741	160	0.6902	206	0.4779	252	0.8999
23	0.1057	69	**0.0000**	115	0.5810	161	0.9090	207	0.6449	253	0.9662
24	0.7518	70	0.2237	116	0.8878	162	0.8805	208	0.8118	254	0.8389
25	0.6024	71	0.7164	117	0.8180	163	0.6761	209	0.9787	255	0.7115
26	**0.0000**	72	0.8044	118	0.5312	164	0.4716	210	0.8639	256	0.5841
27	0.3976	73	0.3438	119	0.2444	165	0.2670	211	0.7078	257	0.4568
28	0.9564	74	**0.0000**	120	**0.0000**	166	0.0626	212	0.5519	258	0.3294
29	0.3103	75	0.1956	121	0.1820	167	**0.0000**	213	0.3958	259	0.2021
30	**0.0000**	76	0.6562	122	0.4688	168	0.1195	214	0.2398	260	0.0748
31	0.0436	77	0.8910	123	0.7556	169	0.3239	215	0.0838		
32	0.6897	78	0.4605	124	0.9604	170	0.5284	216	**0.0000**		
33	0.6645	79	0.0301	125	0.6923	171	0.7330	217	0.1361		
34	0.0184	80	**0.0000**	126	0.4242	172	0.9374	218	0.2922		
35	**0.0000**	81	0.1090	127	0.1562	173	0.8674	219	0.4481		
36	0.3355	82	0.5395	128	**0.0000**	174	0.6762	220	0.6042		
37	0.9816	83	0.9699	129	0.0396	175	0.4851	221	0.7602		
38	0.3724	84	0.6258	130	0.3077	176	0.2940	222	0.9162		
39	**0.0000**	85	0.2235	131	0.5758	177	0.1029	223	0.9325		
40	0.6276	86	**0.0000**	132	0.8438	178	**0.0000**	224	0.7867		
41	0.7266	87	0.3742	133	0.8954	179	0.1326	225	0.6409		
42	0.0805	88	0.7765	134	0.6449	180	0.3238	226	0.4951		
43	**0.0000**	89	0.8330	135	0.3944	181	0.5149	227	0.3493		
44	0.2734	90	0.4570	136	0.1439	182	0.7060	228	0.2035		
45	0.9195	91	0.0810	137	**0.0000**	183	0.8971	229	0.0577		

Tabela 6.3: *Conteúdo da memória Valor_ Filtro_ ROM.*

6.2.3 Logaritmo da Energia

Para a etapa do cálculo do Logaritmo, é utilizado o algoritmo CORDIC (*COordinate Rotation DIgital Computer*) [153], que fornece um método iterativo para a rotação de vetores em diferentes ângulos ou alinhamento com um dos eixos coordenados, usando apenas deslocamentos e somas. Estas operações podem ser realizadas em sistemas de coordenadas lineares, circulares e hiperbólicas.

Para o cálculo do logaritmo são necessárias coordenadas hiperbólicas. O algoritmo utiliza rotações em ângulos elementares de valor conhecido ($tanh^{-1}(2^{-i})$) armazenadas em uma memória. A iteração básica ou microrotação em coordenadas hiperbólicas é:

$$\begin{aligned} x_{i+1} &= x_i - m.y_i.\delta_i.2^{-i} \\ y_{i+1} &= y_i + m.x_i.\delta_i.2^{-i} \\ z_{i+1} &= z_i - \delta_i.arctgh(2^{-i}), \end{aligned} \tag{6.10}$$

sendo (x_0, y_0) as coordenadas iniciais do vetor e a coordenada z acumula o valor do ângulo. O coeficiente δ_i indica a direção de cada microrotação.

O domínio de convergência do CORDIC está limitado pelo intervalo $[\frac{-\pi}{2}, \frac{\pi}{2}]$, devido ao uso de 2^0 para tangente hiperbólica na primeira iteração [153]. Para números fora deste intervalo, aplica-se o algoritmo para o valor escalonado. A relação entre o arco tangente hiperbólico de um número β e o logaritmo natural do mesmo é:

$$ln(\beta) = 2.arctgh\left(\frac{\beta - 1}{\beta + 1}\right) \tag{6.11}$$

Devido à divergência do arco tangente hiperbólico no ponto 1, o intervalo para os valores de cálculo de ver $0.5 \leq \beta < 1$. Assim, os valores dentro do intervalo $1 < \beta < 0.5$ são multiplicados ou divididos por 2 até satisfazer a regra anterior. De maneira analítica:

$$ln(\beta) = ln(\beta^{'}.2^n) = ln(\beta^{'}) + n.ln(2), \tag{6.12}$$

sendo n um número inteiro positivo quando $\beta < 0.5$, zero quando $0.5 \leq \beta < 1$ ou um número inteiro negativo quando $\beta > 1$.

O sistema proposto está composto de duas etapas de processamento e é apresentado na Figura 6.21. Inicialmente, o valor de entrada é analisado e, se necessário, realiza-se o escalonamento do mesmo. Em seguida, o algoritmo CORDIC é aplicado ao valor residual e o resultado é somado com o fator proveniente da etapa de escalonamento.

Na Figura 6.22, é apresentada a arquitetura que realiza o escalonamento. Inicialmente, o sinal de entrada é armazenado em um registrador de deslocamento de 16 bits (*In_ Shift_ Reg*).

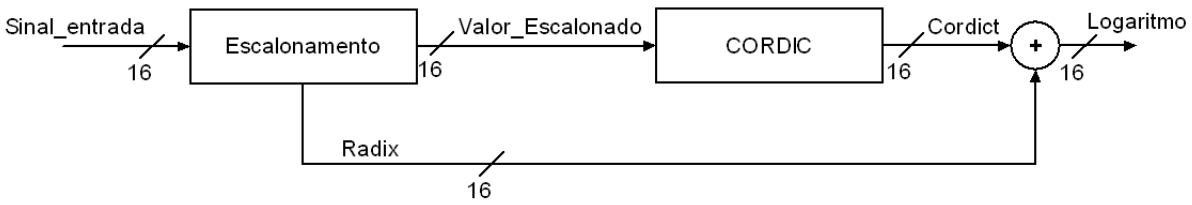


Figura 6.21: *Diagrama de blocos do processador logaritmo.*

A cada pulso de *clock*, um bit vai sendo armazenado no registrador *Out_ Shift_ Reg*. O deslocamento é desativado, quando o escalonamento é concluído, ativando-se o sinal *Done*.

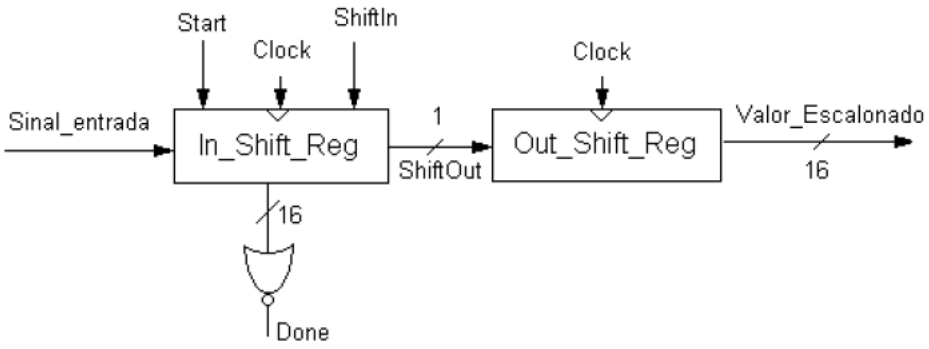


Figura 6.22: *Cálculo do valor escalonado.*

Na Figura 6.23, é apresentado o cálculo do fator de escalonamento, *Radix*. O fator de escalonamento é $n.ln(2)$, sendo n o número de vezes que é realizado o deslocamento do valor inicial do sinal de entrada. Para melhor precisão, é utilizado um barramento de dados interno de 20 bits, sendo externamente aproximado para 16 bits.

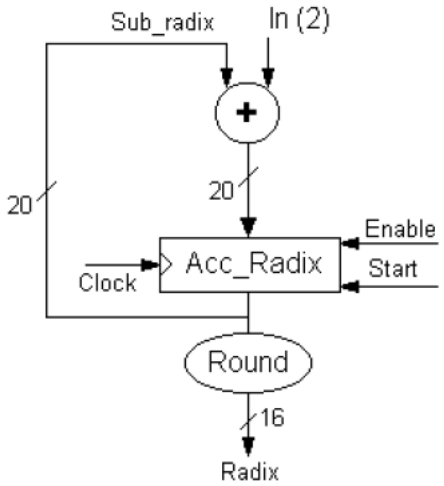


Figura 6.23: *Cálculo do fator de escalonamento.*

A arquitetura do módulo CORDIC projetada é apresentada nas Figuras 6.24 e 6.25. Inicialmente, o valor escalonado é selecionado pelos multiplexadores L e o valor 1 é selecionado pelos multiplexadores P. Com estes valores é executada uma soma ou subtração e o resultado

é armazenado nos registradores Reg_X e Reg_Y. Nas seguintes iterações, os multiplexadores L selecionam os conteúdos dos registradores Reg_X e Reg_Y e os multiplexadores P selecionam os conteúdos dos registradores $shift\_x$ e $shift\_y$. Depois disso, o algoritmo para cálculo do logaritmo é executado e o valor final de $ln(\beta')$ é gerado para ser somado ao valor previamente calculado por escalonamento.

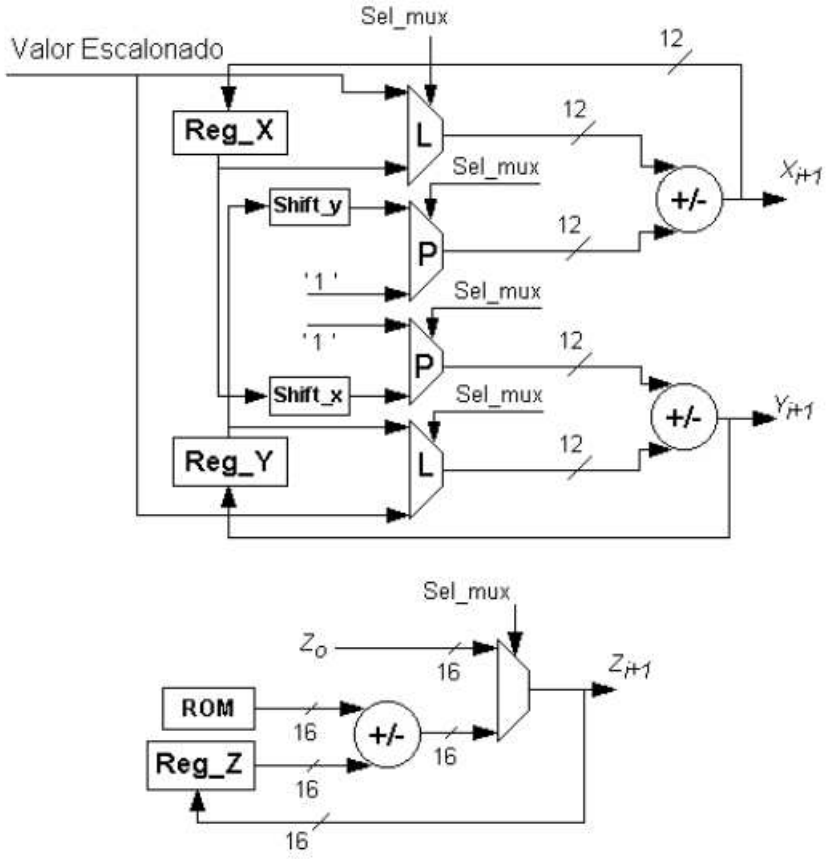


FIGURA 7.13 - Parte operativa da função CORDIC.

Figura 6.24: *Parte operacional do módulo CORDIC.*

Utilizou-se uma representação numérica em ponto-fixo com 16 bits de precisão e representação em complemento de dois. Internamente, trabalhou-se em algumas etapas com 20 bits para aumentar a precisão.

6.2.4 Transformada Inversa Cosseno (IDCT)

A DCT é aplicada ao conjunto de valores provenientes do módulo do logaritmo e calcula a seguinte operação:

$$C_m = \beta . \sum_{j=0}^{N-1} cos\left(m\frac{\pi}{N}(j+0,5)\right).f_j, \quad \beta = \begin{cases} \sqrt{\frac{1}{N}}, & se \quad m=0 \\ \sqrt{\frac{2}{N}}, & se \quad m>0 \end{cases} \tag{6.13}$$

sendo C_m a transformada cosseno da função f_j. N é o número de amostras das quais é

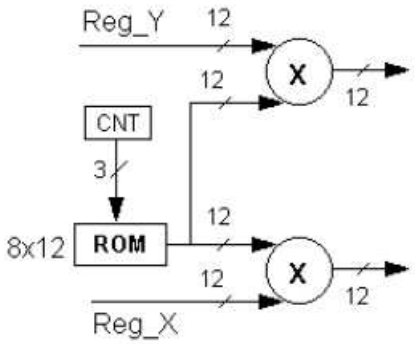


Figura 6.25: *Dispositivos Shift_ X e Shift_ Y.*

calculada a transformada cosseno. Neste caso, fj corresponde ao logaritmo das 36 saídas do banco de filtros triangulares.

A DCT é calculada de maneira similar à dos filtros triangulares, utilizandoo um circuito de multiplicação e acumulação. São calculados os 36 produtos da Equação 6.13, os quais são acumulados para obter o resultado final. Na Figura 6.26, é apresentada a parte operacional da função DCT. O conjunto de valores da função cosseno dentro da transformada foi armazenado em uma memória ROM. Alguns registradores *St0, St1* e *St2* foram utilizados para implementar o *pipeline* do circuito.

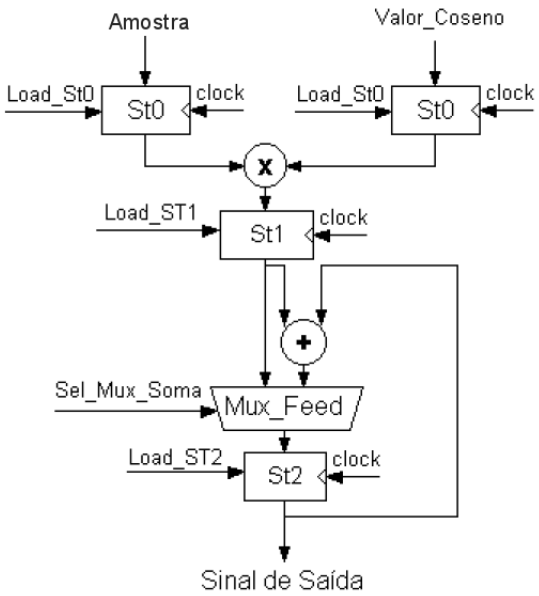


Figura 6.26: *Parte operacional da DCT.*

Ao processar o primeiro quadro, o multiplexador *Mux_Feed* seleciona o cálculo do primeiro produto e armazena-o no acumulador *St2*. Os produtos que são calculados depois são selecionados por *Mux_ Feed*, para serem somados com o valor inicial do acumulador *St2*, até completar os 36 produtos correspondentes ao primeiro quadro. O processo é repetido com o primeiro produto do seguinte quadro e depois com os valores restantes do quadro. Este processo continua até completar o total dos quadros.

Na Figura 6.27, é apresentada a conexão entre as memórias utilizadas. O sinal de entrada, formado pelos 36 logaritmos referentes ao banco de filtros triangulares, é armazenado temporariamente em uma memória RAM RW de dupla porta de acesso. A escrita nesta RAM é realizada durante a borda de descida do clock e a leitura durante a borda de subida. Um contador de 6 bits, *Up_ Counter6*, é utilizado para endereçar esta memória de 0 a 35. O sinal *Reset* deste contador também é utilizado para controlar *Mux_ Feed*. A memória ROM de cossenos é endereçada por um contador de 11 bits, *Up_ Counter11*, que varia de $0..36^2-1$.

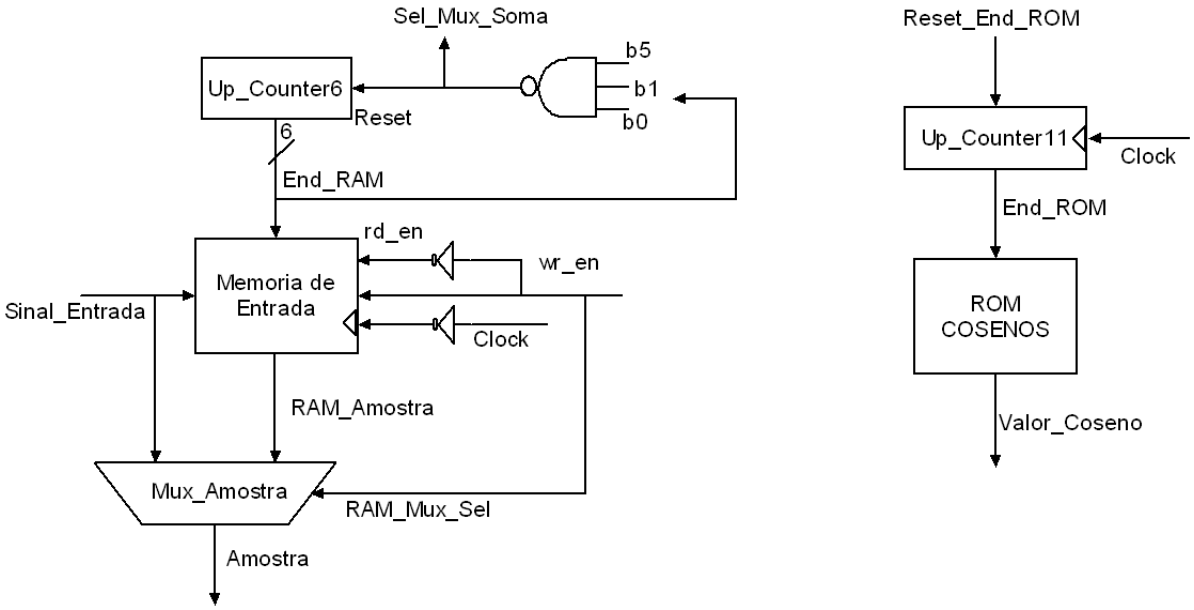


Figura 6.27: *Endereçamento das memórias do módulo DCT.*

6.2.5 Delta e 2Delta MFCC

Como foi explicado anteriormente no Capítulo 2, para melhor refletir as mudanças dinâmicas dos MFCC no tempo, a primeira e segunda derivadas em tempo podem ser utilizadas. Estas podem ser computadas calculando a diferença entre coeficientes de m índices no passado e no futuro do tempo levado em consideração. A primeira e segunda derivadas são obtidas a partir das Equações 6.14 e 6.15, respectivamente, repetidas aqui por conveniência:

$$\Delta c_{mel}[n] = c_{mel}[n+m] - c_{mel}[n-m] \tag{6.14}$$

$$\Delta\Delta c_{mel}[n] = \Delta c_{mel}[n+m] - \Delta c_{mel}[n-m], \tag{6.15}$$

sendo n o índice o coeficiente em questão e m, comumente no intervalo $1 \leq m \leq 4$ [59].

Com a utilização das derivadas, o vetor de características utilizado neste trabalho é

composto pelos parâmetros MFCC, seguidos de suas primeira e segunda derivadas, conforme ilustrado na Figura 6.28.

Figura 6.28: *Vetor de características incluindo primeira e segunda derivadas de MFCC.*

Na Figura 6.29, é ilustrado graficamente o processo de obtenção da primeira e segunda derivada dos coeficientes MFCC calculados anteriormente.

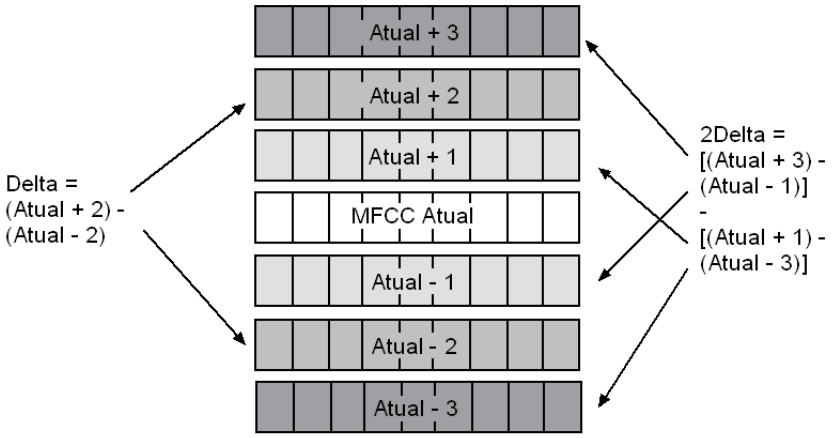


Figura 6.29: *Obtenção da primeira e segunda derivadas de MFCC.*

Para o cálculo das derivadas são utilizadas uma memória ROM e uma memória RAM RW. A RAM *RAM_COEFS*, de 27 palavras armazena os coeficientes MFCC obtidos após a etapa da Tranformada Cosseno, como também as derivadas que serão calculadas. Já a ROM *ROM_POS_MFCC*, de 9 palavras de 24 bits cada, armazena as posições dos coeficientes MFCC que serão utilizados no cálculo das derivadas. Como foi ilustrado na Figura 6.29, para obtenção das derivadas são utilizados os coeficientes de uma a três posições anteriores e posteriores ao atual. Assim, *ROM_POS_MFCC* armazena em cada endereço, um para cada coeficiente (0 a 8), o endereço em *RAM_COEFS* dos coeficientes necessários para o cálculo da derivada correspondente. O conteúdo de *ROM_POS_MFCC* é ilustrado na Tabela 6.4.

End	Ap2	Aa2	Ap3	Aa1	Ap1	Aa3
0	2	7	3	8	1	6
1	3	8	4	0	2	7
2	4	0	5	1	3	8
3	5	1	6	2	4	0
4	6	2	7	3	5	1
5	7	3	8	4	6	2
6	8	4	0	5	7	3
7	0	5	1	6	8	4
8	1	6	2	7	0	5

Tabela 6.4: *Conteúdo da memória ROM_POS_MFCC.*

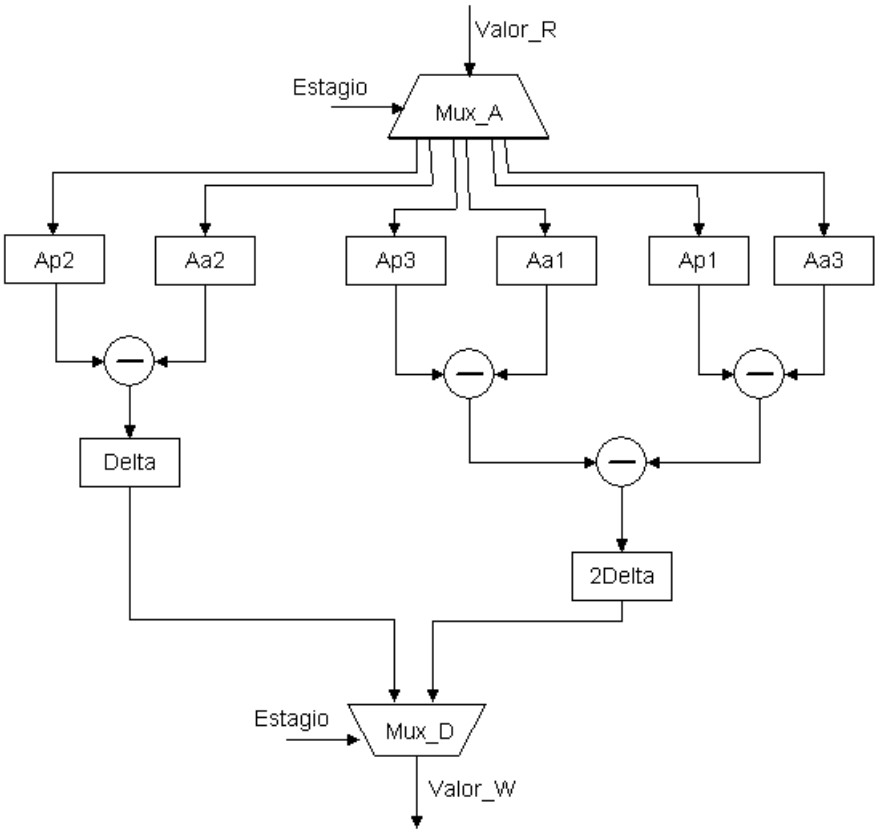


Figura 6.30: *Parte operacional para cálculo da primeira e segunda derivadas de MFCC.*

Na Figura 6.30, é ilustrada a parte operacional do módulo de cálculo das derivadas. O processo é realizado por meio de um *pipeline* de oito estágios:

1. No primeiro estágio, o coeficiente duas posições posteriores ao atual é colocado no registrador *Ap2*. Ainda neste estágio, para é realizada a escrita em *RAM_COEFS* da última segunda derivada calculada, caso este não seja o primeiro coeficiente MFCC.

2. No segundo estágio, o coeficiente duas posições anteriores ao atual é colocado no registrador *Aa2*.

3. No terceiro estágio, é realizada a subtração entre os valores dos registradores *Ap2* e *Aa2*, cujo resultado é armazenado no registrador *Delta*. Ainda neste estágio, o coeficiente três posições posteriores ao atual é colocado no registrador *Ap3*.

4. No quarto estágio, o valor do registrador Delta é escrito em *RAM_COEFS*. Simultaneamente, o coeficiente uma posição anterior ao atual é colocado no registrador *Aa1*.

5. No quinto estágio, o coeficiente uma posição posterior ao atual é colocado no registrador *Ap1*.

6. No sexto estágio, o coeficiente três posições anteriores ao atual é colocado no registrador *Aa3*.

7. No sétimo estágio, são realizadas as subtrações entre os valores dos registradores *Ap3* e *Aa1*, e entre os dos registradores *Ap1* e *Aa3*.

8. No oitavo estágio, é realizada a subtração entre os resultados das duas subtrações anteriores e o resultado desta é armazenado no registrador 2Delta.

Na Figura 6.31, é ilustrado como é realizado o endereçamento das memórias *ROM_POS_MFCC* e *RAM_COEFS*, assim como o controle de estágios do processamento.

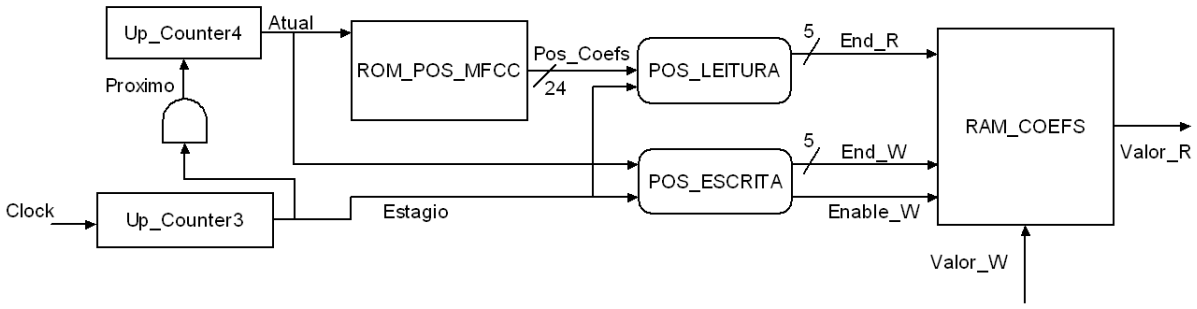


Figura 6.31: *Controle do endereçamento das memórias e estágios do pipeline para cálculo da primeira e segunda derivadas de MFCC.*

O contador *Up_Counter4* é responsável pelo endereçamento de *ROM_POS_MFCC* que fica posicionada no coeficiente atual. O coeficiente *Atual* só é incrementado a cada oito estágios, correspondentes ao *pipeline* descrito anteriormente. *Up_Counter3* faz o controle do estágio atual. O módulo *POS_LEITURA* recebe a palavra de 24 bits de *ROM_POS_MFCC* e a cada estágio pega os 4 bits correspontes ao endereço do coeficiente a ser utilizado no estágio atual. Vale salientar que estes 4 bits são completados com zero para formar o tamanho correto do endereço utilizado por *RAM_COEFS*. O módulo *POS_ESCRITA* juntamente com *Mux_D* também utiliza o estágio atual para controlar o momento exato de escrever os valores de Delta e 2Delta em *RAM_COEFS*. A lógica utilizada por *POS_ESCRITA* e *Mux_D* é apresentada na Figura 6.32. É importante observar que para o último coeficiente, será necessário um ciclo de clock a mais apenas para efetuar a escrita do último valor armazenado em *2Delta*.

```
1   Se Estagio = 0 e Atual > 0
2     Valor_W = 2Delta
3     End_W = (Atual -1) + 18
4     Enable_W = 1
5   Senao Se Estagio = 3
6     Valor_W = Delta
7     End_W = Atual + 9
8     Enable_W = 1
9   Senao
10    Enable_W = 0
```

Figura 6.32: *Lógica do componente POS_ESCRITA e Mux_D.*

É importante destacar que, caso se deseje utilizar outro número de coeficientes, maior ou menor que nove, basta modificar o conteúdo de *ROM_POS_MFCC*, de modo que uma quantidade maior ou menor de valores provenientes da DCT seja considerada.

6.3 Cálculo das Gaussianas

Depois de extraídas as características do sinal de voz, a próxima etapa consiste em identificar a qual modelo treinado estas correspondem. Para isso, o primeiro passo é calcular a probabilidade que essas caracteristicas tenham sido geradas por cada um dos modelos conhecidos. Para um sistema que utiliza HMM contínuo, como foi exposto no Capítulo 3, a probabilidade dos modelos acústicos é expressa por uma mistura finita de M gaussianas multidimensionais. No intuito de simplificar a implementação em um sistema embarcado, como foi explicado no Capítulo 5, essa probalidade é convertida em custo e pode ser calculado por meio da Equação 6.16.

$$c_j = \min_{m=1}^{M} \left(\alpha_{j,m} + \sum_{k=1}^{K} \frac{(x_k - \mu_{j,m,k})^2}{2\sigma_{j,m,k}^2} \right), \tag{6.16}$$

sendo j o índice do modelo acústico atual e M o número de misturas utilizadas no modelo, que neste trabalho corresponde a 8 misturas gaussianas. x_k são os coeficientes MFCC, juntamente com suas primeira e segunda derivadas. Os modelos acústicos são definidos por suas médias (μ), variância (σ^2) e coeficientes de peso de cada mistura (w).

De modo a simplificar ainda mais o cálculo do módulo das gaussianas, a variância σ^2 é substituída por uma nova variável v' que é descrita segundo a Equação 6.17. Armazenando-se este valor pré-calculado ao invés da variância, o cálculo do custo de cada vetor de característica para um determinado modelo acústico, é dado pela Equação 6.18.

$$v' = \frac{1}{\sqrt{2\sigma^2}} \tag{6.17}$$

$$c_j = \min_{m=1}^{M} \left(\alpha_{j,m} + \sum_{k=1}^{K} ((\mu_{j,m,k} - x_k).v'_{j,m,k})^2 \right) \tag{6.18}$$

Na Figura 6.33, é apresentado o diagrama de blocos do módulo para cálculo das gaussianas, baseado na Equação 6.18.

O bloco *Artimetica* calcula a seguinte parte da Equação 6.18:

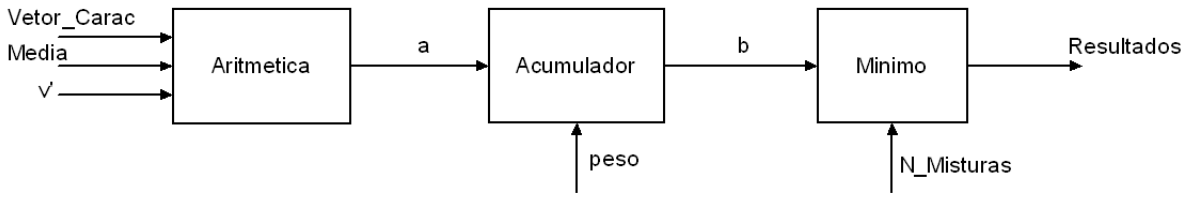


Figura 6.33: *Diagrama de bloco do módulo das Gaussinas.*

$$a = ((\mu_{j,m,k} - x_k).v'_{j,m,k})^2) \tag{6.19}$$

A parte operacional de *Artimetica* é apresentada na Figura 6.34.

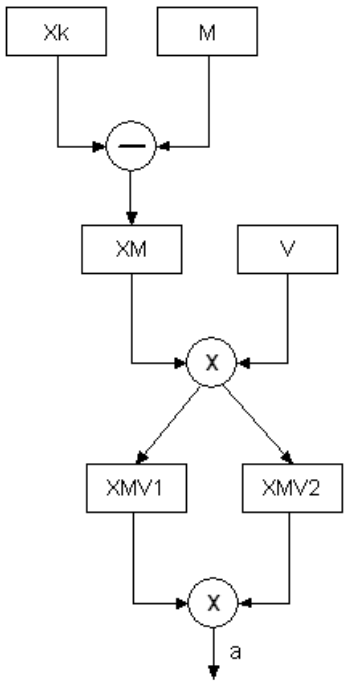


Figura 6.34: *Diagrama de bloco do módulo das Gaussianas.*

O bloco *Acumulador* calcula a soma dos 27 valores (MFCC + derivadas) de a, que correspondem a uma mistura, e soma ao peso daquela mistura. O último bloco, *Minimo*, seleciona o menor valor de b em cada modelo, levando em consideração o número de misturas daquele modelo ($N\_Misturas$). As entradas do bloco *Artimetica* são de 16 bits em ponto-fixo e os demais valores 32 bits em ponto-fixo.

Também foi projetado um esquema de *prunning* para reduzir o número de modelos a serem analisados pelo módulo das Gaussianas. Na Figura 6.35, é apresentado o diagrama de blocos para implementação do *prunning*, na qual o bloco Gauss é o mesmo descrito na Figura 6.33.

Como foi definido no Capítulo 5, os modelos gerados durante a etapa de treinamento são armazenados em uma memória externa, aqui referenciada por $Mem\_Modelos$. Foram

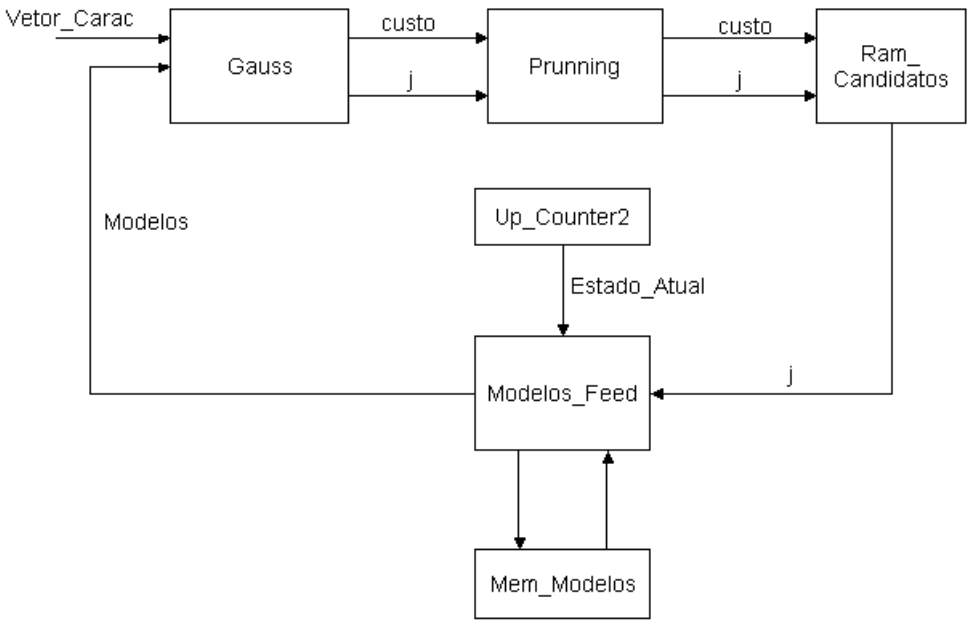


Figura 6.35: *Prunning para o módulo das Gaussianas.*

utilizados HMM de três estados e o trifone como unidade fonética. Cada trifone é formado por uma sequência de três estados, que representam cada um dos modelos j. Estes trifones são ordenados na memória pela sequência de estados (modelos) que o representa, como é ilustrado na Tabela 6.5. Essa ordenação agiliza a seleção, por *Modelos_ Feed*, dos modelos candidatos ao próximo estado. Assim, para cada vetor de características obtido, é calculado o custo associado ao mesmo, naquele estado, a cada um dos modelos conhecidos pelo sistema.

O módulo *Prunning* recebe a saída de Gauss, que representa o custo de cada modelo j analisado, e verifica se este está acima ou abaixo de um limiar definido. Se o custo estiver acima do limiar, o modelo é descartado como candidato ao próximo estado. Senão, o índice do modelo (j), o estado atual e o custo são armazenados em uma RAM RW, *RAM_ Candidatos*.

O módulo *Modelos_ Feed* tem a função de alimentar o módulo *Gauss* com os modelos candidatos, com base no estado atual. *RAM_ Candidatos* é lida por *Modelos_ Feed* que com base nos valores de j seleciona os modelos que serão analisados por Gauss no próximo estado.

Para isso, a partir do segundo estado, *Modelos_ Feed* localiza em *Mem_ Modelos*, o primeiro trifone cujo estado anterior ao atual seja igual a j e vai repassando para *Gauss*, até que este seja maior que j. Por exemplo, no primeiro estado (Estado_atual=0) serão repassados para *Gauss*, todos os modelos de Mem_ Modelos. No segundo estado (Estado_atual=1), serão repassados para Gauss apenas os modelos, cujo Estado1 em *Mem_ Modelos*, seja igual aos valores de j selecionados e armazenados em RAM_Candidatos. Já no terceiro estado, serão repassados para Gauss apenas os modelos, cujo Estado2 seja igual aos valores de j e Estado1 igual ao j anterior. A lógica completa de *Modelos_ Feed* é apresentada na Figura 6.36.

Trifone	Estado1	Estado2	Estado3
...	...	...	...
a a m	121	160	212
a a m	121	160	212
a a n	121	160	212
a a n	121	160	212
a a nn	121	182	221
a a o	121	192	224
a a on	121	192	238
a a oo	121	192	238
a a p	121	198	240
a a p	121	198	240
...	...	...	...
a d b	129	164	227
a d b	129	164	228
a d d	129	165	227
a d d	129	165	228
a d d	129	166	227
a d dd	129	166	228
a d dd	129	169	227
a d dd	129	169	228
a d dd	129	169	229
a d e	129	174	230
...	...	...	...

Tabela 6.5: *Ordenação da memória Mem_ Modelos.*

```
1   Se Estado_atual = 0
2     Repassa todos os modelos para Gauss
3
4   Se Estado_atual = 1
5     Para todos os j de Ram_Candidatos
6       Localiza primeiro trifone cujo Estado1 = j
7       Enquanto Estado1 = j
8         Repassa os modelos de indice Estado2
9     Estado_Ant = j
10
11  Se Estado_atual = 2
12    Para todos os j de Ram_Candidatos
13      Localiza primeiro trifone cujo Estado1 = Estado_Ant e Estado2 = j
14      Enquanto Estado2 = j
15        Repassa os modelos de indice Estado3
```

Figura 6.36: *Lógica do componente Modelos_ Feed.*

6.4 Algortimo de Viterbi combinado ao Modelo Linguístico

A última etapa do processo de reconhecimento consiste na utilização das probabilidades, neste trabalho custos, obtidas na etapa das Gaussianas, para definir a mais provável sequência de estados do HMM e, consequentemente, das palavras pronunciadas. Como foi descrito anteriormente, para encontrar tal sequência é utilizado o algoritmo de Viterbi.

Neste trabalho, os HMMs são utilizados para representar trifones. Os trifones são combinados para formar palavras e, essas combinadas para formas as sentenças. Sendo assim, para o reconhecimento da sentença pronunciada, é necessário encontrar a sequência de estados de HMMs diferentes, o que implica em algumas complicações. Neste caso, é necessário considerar se está sendo realizada uma transição entre sons atômicos (senones) em um mesmo trifone, ou de um trifone para outro dentro de uma mesma palavra (intrapalavra), ou ainda, entre o trifone final de uma palavra para o trifone inicial de outra palavra (interpalavra). No último caso, a utilização do Modelo Linguístico (ML) torna-se um elemento essencial para aumentar a acurácia do processo de reconhecimento.

Diante do que foi exposto, é necessário adaptar o algoritmo de Viterbi para considerar as probabilidades fornecidas pelo ML. Como foi exposto no Capítulo 5, é utilizado o logaritmo dos parâmetros do modelo HMM no algoritmo de Viterbi, de modo a evitar *underflow*. As equações adaptadas são repetidas aqui por conveniência:

1. Inicialização:

$$\begin{aligned} \delta_1(i) = \pi_i + b_i(O_1) \quad &, 1 \leq i \leq N, \\ \psi_1(i) &= 0. \end{aligned} \tag{6.20}$$

2. Recursividade:

$$\begin{aligned} \delta_t(j) &= \max_{1\leq i\leq N}[\delta_{t-1}(i) + a_{ij}] + b_j(O_t), \\ \psi_t(j) &= \underset{1\leq i\leq N}{argmax}[\delta_{t-1}(i) + a_{ij}], \quad 2 \leq t \leq T, \quad 1 \leq j \leq N. \end{aligned} \tag{6.21}$$

3. Término:

$$\begin{aligned} P^* &= \max_{1\leq i\leq N}[\delta_T(i)], \\ q_T^* &= \underset{1\leq i\leq N}{argmax}[\delta_T(i)]. \end{aligned} \tag{6.22}$$

4. Sequência de estados ótima

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \quad , t = T-1, T-2, \ldots, 1. \tag{6.23}$$

Como pode ser observado nas equações apresentadas, a probabilidade acumulada da sequência de estados mais provável ao alcançar o estado s_j, para um determinada palavra w é obtida por:

$$\delta_t(s_j; w) = \max_i[\delta_{t-1}(s_i; w) + a_{ij}] + b(O_t; s_j, w) \tag{6.24}$$

Para as transições interpalavras, a Equação 6.24 é adaptada para incluir a probabilidade fornecida pelo ML, como é exibido em 6.25. Sendo $p(w|v)$ a probabilidade do ML, que informa a probabilidade de transição da palavra de origem v para a palavra seguinte w. s_f indica o estado final e s_0 o estado pseudo-inicial. É importante destacar que, assim como para os demais parâmetros do HMM, é utilizado o logaritmo da probabilidade do ML.

$$\delta_t(s_0; w) = \max_v[log(p(w|v)) + \delta_t(s_f; v)] \tag{6.25}$$

A parte operacional que implementa as equações descritas, é apresentada na Figura 6.37. O componente *Viterbi_ Feed* é responsável por fornecer dados para os demais componentes desta etapa, assim como manter as informações sobre a sequência de estados mais provável. Para o cálculo de $\delta_t(j)$, são utilizados três componentes *Processa_ Estado* (Figura 6.38), sendo um para cada estado.

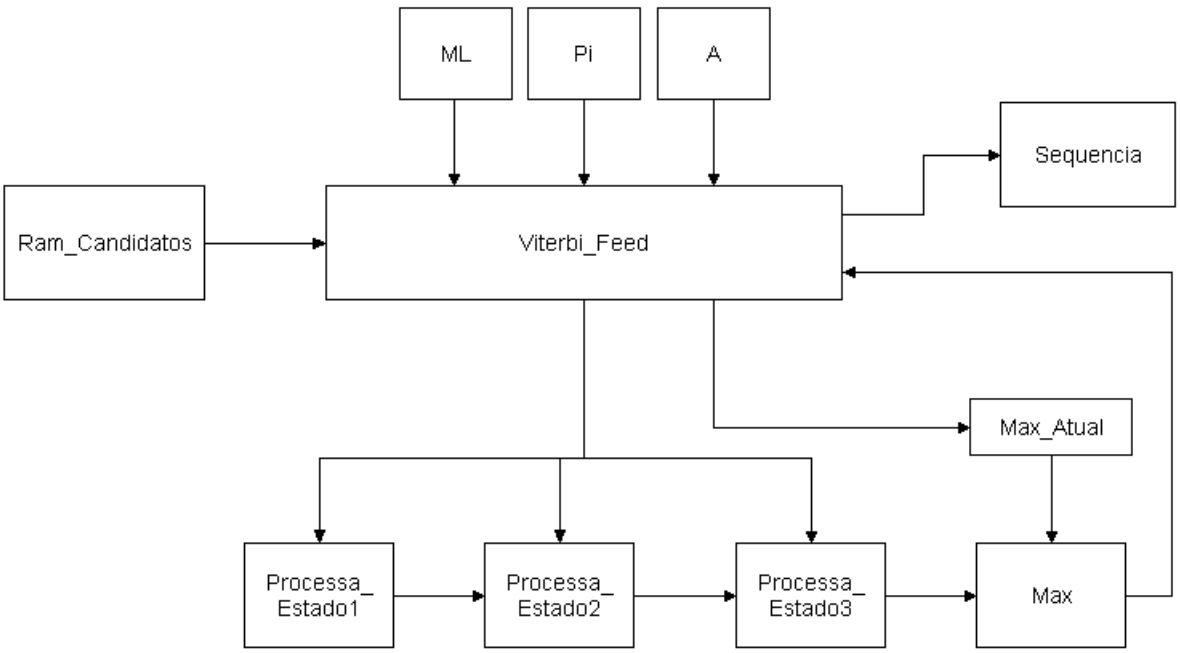


Figura 6.37: *Parte operacional Viterbi.*

Para fornecer os dados necessários para os demais componentes deste módulo, *Viterbi_ Feed* utiliza as informações armazenadas em *RAM_ Candidatos*, apresentada no módulo das gaussianas. Como foi explicado anteriormente, nesta memória estão armazenados todos os es-

tados, cuja probabilidade de emissão estava acima do limiar definido pelo sistema, como também o instante t no qual foi emitido. Com base nessas informações e o estado atual, *Viterbi_Feed* fornece adequadamente, para cada um dos componentes *Processa_Estado*, a probabilidade inicial (π_j), a probabilidade de transição (a_{ij}) e a probabilidade de emissão ($b_j(O_t)$).

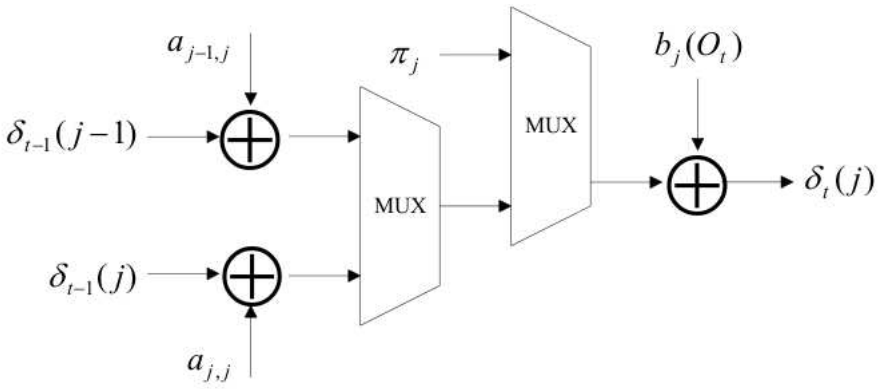


Figura 6.38: *Componente Processa_Estado.*

Cada componente *Processa_Estado* é composto por três somadores e dois multiplexadores que funcionam como seletores. Inicialmente é calculado o valor de $\delta_{t-1}(j-1) + a_{j-1,j}$ e $\delta_{t-1}(j) + a_{j,j}$. Em seguida, é escolhido o menor valor entre eles, ao qual é somado o valor de $b_j(O_t)$, obtendo assim $\delta_t(j)$. No estado inicial, só é necessário comparar os valores de $\delta_{t-1}(j) + a_{j,j}$ e π_j.

Como foi descrito anteriormente, outra função de *Viterbi_Feed* é manter a sequência de estados mais provável. A cada saída do terceiro componente *Processa_Estado*, o valor é comparado com *Max_Atual*, que armazena a maior probabilidade até o modelo atual. Caso tenha sido encontrada uma probabilidade maior, *Viterbi_Feed* atualiza *Max_Atual* e salva a nova sequência de estados.

Depois de identificar a melhor sequência de estados, um trifone é obtido. Com base nos modelos acústicos, gerados durante o treinamento, é possível saber se este trifone corresponde a uma transição interpalavra. Se este for o caso, a única alteração no procedimento descrito, é para o valor da probabilidade inicial. Esta será dada pelo maior valor entre a soma da probabilidade acumulada do último estado da palavra anterior (atual) com a probabilidade fornecida pelo ML, e a probabilidade do primeiro estado da palavra candidata. Esse processamento deve ser realizado para cada uma das palavras candidatas a sucederem a atual.

O ML é armazenado por ordem da palavra antecessora (Figura 6.39). Assim, se torna mais fácil identificar as palavras candidatas a sucederem a atual. Com base nas palavras candidatas, é possível identificar qual o próximo estado candidato. Isso porque, como foi descrito anteriormente, palavras consistem de trifones conectados, ou seja, HMMs conectados, ou ainda, estados conectados. Assim, *Viterbi_Feed* fornece para cada um dos demais componentes, os dados necessários para calcular a probabilidade de transição do último

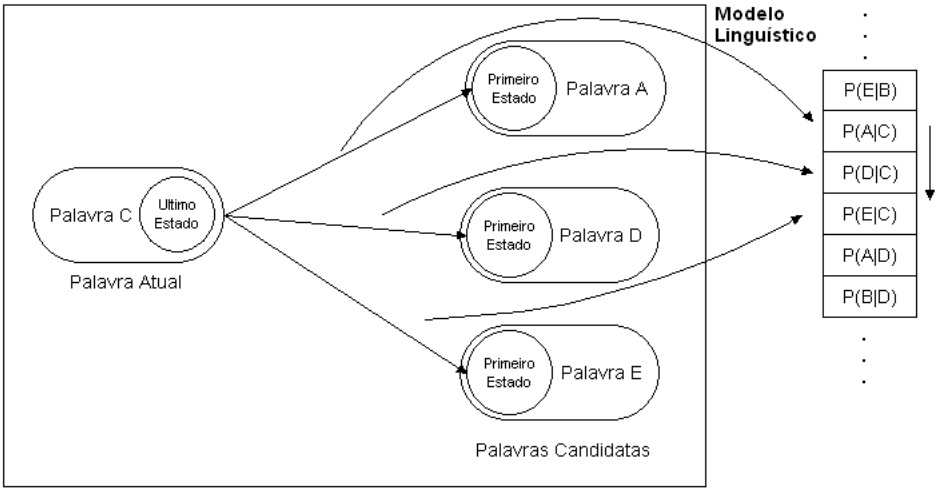


Figura 6.39: *Transição intrapalavra.*

estado da palavra atual, para o primeiro estado de cada uma das palavras candidatas.

6.5 Considerações Gerais

Neste capítulo foi apresentada a arquitetura desenvolvida do sistema para reconhecimento de fala contínua, utilizando HMM contínuo e coeficientes mel-cepstrais. A arquitetura contempla desde a etapa inicial de pré-processamento do sinal de voz, até a etapa final que identifica a sequência de fonemas da sentença pronunciada.

Na etapa de pré-processamento foram projetadas as arquiteturas para as funções de prê-enfase e janelamento. Foram realizadas simplificações matemáticas nas equações utilizadas, e valores que podiam ser pré-calculados em *software* foram armazenados em memórias ROM, evitando assim o cálculo dos mesmos em tempo de execução.

Na etapa de extração de características, foi modelada uma arquitetura para cálculo dos coeficientes mel-cepstrais (MFCC), juntamente com suas primeiras e segundas derivadas. A quantidade de coeficientes e filtros utilizados foram definidos com base nos experimentos realizados em uma etapa anterior deste trabalho. Aqui também foram identificados valores que podiam ser pré-calculados, os quais foram armazenandos em memórias ROM, evitando a necessidade de um *hardware* extra para cálculo dos mesmos em tempo de execução. Outra vantagem da arquitetura desenvolvida é que, com pequenas modificações, é possível utilizar diferentes números de coeficientes.

Na etapa de cálculo das gaussianas, foi realizada uma simplificação matemática na equação correspondente, o que diminui consideravelmente a quantidade de operações complexas a serem executadas, como multiplicações e exponenciais (vide Capítulo 5). Além disso, foi definido um esquema de *prunning* que diminui o número de modelos de referência a serem analisados pelo sistema. Mais uma vez, valores pré-calculados foram utilizados e armazenados em ROMs.

Por fim, na etapa correspondente ao algoritmo de Viterbi, foram realizadas simplifica-

ções matemáticas nas equações utilizadas, e que reduziram operações mais complexas como multiplicações, a simples somas, que requerem um *hardware* bem mais simplificado. Outra modificação relevante nesta etapa, foi a incorporação das probabilidades fornecidas pelo Modelo Linguístico (ML) no algortimo.

Capítulo 7

Considerações Finais e Sugestões para Trabalhos Futuros

A possibilidade de realizar a comunicação homem-máquina com a utilização da voz torna essa interação mais fácil e produtiva, uma vez que além de ser a forma mais natural de comunicação humana, permite que as mãos e os olhos dos usuários fiquem disponíveis para outras tarefas. Diante do avanço tecnológico e consequente surgimento de equipamentos eletrônicos cada vez mais sofisticados, a possibilidade de permitir essa interação a partir da voz tem sido objeto de grande interesse, tanto do meio acadêmico quanto dos fabricantes de tais equipamentos. Pesquisas recentes na área de Processamento Digital de Sinais de Voz (PDSV) têm permitido o desenvolvimento de sistemas de Reconhecimento de Fala bastante eficientes. Entretanto, requisitos de processamento ainda dificultam a implementação desses sistemas em dispositivos com pequeno poder computacional, como celulares, *palmtops* e eletrodomésticos.

Tendo em vista o vigente interesse pelo desenvolvimento de aplicações de PDSV para sistemas embarcados e também as dificuldades enfrentadas neste tipo de projeto, torna-se clara a importância da definição e implantação de ferramentas de *hardware* e *software* que otimizem essa produção. Muitos trabalhos podem ser encontrados para reconhecimento de palavras isoladas nesse contexto [10, 11, 15], ou ainda reconhecimento de fala contínua para idiomas diferentes do português brasileiro (PB) [26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36]. No entanto, poucos tratam de fala contínua para a língua portuguesa em sistemas embarcados.

As tecnologias mais promissoras na área de reconhecimento de fala, como por exemplo, *Hidden Markov Models* (HMM) utilizam métodos de modelagem estatística que aprendem por exemplos, e devem considerar no treinamento as possíveis variações da fala e o ambiente de utilização do processo do sistema de reconhecimento [91, 131, 133, 138, 139, 140]. Sendo assim, esse tipo de aplicação exige uma quantidade de dados de treinamento representativa, que permita a correta modelagem dos padrões a serem reconhecidos. O efeito causado por variáveis não modeladas ou mal modeladas (tais como diferenças de canal ou microfones, palavras fora do vocabulário, sub-unidades fonéticas mal treinadas) podem comprometer

seriamente o desempenho dos sistemas de reconhecimento de fala. Outro fator que deve ser considerado quando se trata de reconhecimento de fala são as características acústicas e linguísticas de cada idioma. Diferentes idiomas possuem diferentes fonemas, tanto em número quanto em características (duração, intensidade, etc), logo parâmetros e configurações que oferecem altas taxas de reconhecimento em determinado idioma, podem não ser adequados para um outro. Enquanto que, para o idioma inglês existem várias pesquisas e bons resultados, pouco se tem para o idioma português brasileiro [49, 131, 133, 138, 140].

O objetivo principal deste trabalho consistiu na modelagem da arquitetura de um sistema de reconhecimento de fala contínua para o português brasileiro, utilizando *Hidden Markov Models*, de forma a possibilitar sua implementação em um sistema embarcado com recursos computacionais limitados.

Para a construção de um sistema de reconhecimento de fala contínua para o português brasileiro eficiente, como foi dito anteriormente, torna-se necessária a adequação dos modelos acústicos e linguísticos para esse idioma, a partir de testes variando os principais parâmetros utilizados [38, 132, 138, 140]. Para realização desses testes, neste trabalho foi utilizado o *framework* Sphinx, quando foram variados: número de coeficientes MFCC, número de filtros para obtenção dos MFCC e número de estados do HMM. A base de dados utilizada foi desenvolvida em [143], sendo constituída por 200 frases foneticamente balanceadas definidas em [142], pronunciadas por 46 diferentes locutores, sendo 24 do gênero masculino e 22 feminino, de diferentes idades (18 a 60 anos), graus de instrução e cidades do Brasil.

Uma vez que o objetivo deste trabalho era identificar uma configuração que forneça boas taxas de reconhecimento sem exigir muitos recursos computacionais, embora a literatura aponte para o uso de 12 ou 13 coeficientes MFCC [131, 132, 133, 134], foram realizados testes com outros valores abaixo desse valor. O intuito foi avaliar o impacto da utilização de um número reduzido de coeficientes na taxa de reconhecimento, e no tempo de execução para as etapas de treinamento e reconhecimento.

7.1 Contribuições

A partir dos resultados obtidos neste trabalho, pode-se citar as seguintes contribuições:

1. Avaliação dos parâmetros e da configuração de um sistema de reconhecimento de fala contínua utilizando MFCC, HMM e modelagem linguística *N-gram*, diante de uma base de dados foneticamente balanceada e representativa. Foram realizados experimentos com um número reduzido de coeficientes, obtendo-se boas taxas de reconhecimento e assim, reduzindo a quantidade de recursos computacionais necessários.

2. Simplificações matemáticas nas equações utilizadas nas seguintes etapas do processo de reconhecimento:

- Pré-processamento - além de simplificações matemáticas que transformaram operações complexas como multiplicações em simples deslocamentos de bits, valores que podiam ser pré-calculados em *software* foram armazenados em memórias ROM, evitando assim o cálculo dos mesmos em tempo de execução. Tais estratégias permitem a construção de *hardware* bem mais simples e reduzido do que utilizando as abordagens tradicionais.
- Extração de características - foi modelada uma arquitetura para cálculo dos coeficientes mel-cepstrais (MFCC), juntamente com suas primeiras e segundas derivadas. A quantidade reduzida de coeficientes e filtros utilizados foram definidos com base em experimentos também realizados neste trabalho. Aqui também foram identificados valores que podiam ser pré-calculados, os quais foram armazenandos em memórias ROM, evitando a necessidade de um *hardware* extra para cálculo dos mesmos em tempo de execução.
- Cálculo das Gaussianas - a simplificação utilizada, reduziu consideravelmente o número de operações mais complexas como multiplicações e exponenciais necessárias para sua execução. Para o caso de exponenciais envolvendo a constante neperiana (e), estas são totalmente eliminadas.
- Algoritmo de Viterbi - com as simplificações matemáticas as operações de multiplicações são reduzidas à somas. O que provê as vantagens já citadas.

3. Definição de um esquema de *prunning* para a etapa das Gaussianas, que além de reduzir a quantidade de modelos a serem avaliados nesta etapa, também simplificou a etapa do algoritmo de Viterbi. Como foi exposto, esse esquema faz com que ao invés de calcular a probabilidade para cada um dos padrões de referência conhecidos pelo sistema, apenas os modelos candidatos ao próximo estado são avaliados. Essa simplificação se mostra ainda mais relevante, à medida que a quantidade de modelos acústicos, ou seja, o vocabulário, aumenta.

4. Construção de um modelo linguístico para o *corpus* da base de dados, a partir de uma ferramenta que pode ser utilizada na construção de modelos linguísticos para bases com *corpus* ainda maiores. Como foi exposto neste trabalho, a utilização de uma base de dados representativa é fator de grande importância para construção de um sistema de reconhecimento eficiente. Apesar disso, a aquisição de bases de dados para o idioma português brasileiro, não é um processo trivial. A base de dados obtida e utilizada neste trabalho, originalmente, não incluía um Modelo Linguístico. A validação da ferramenta utilizada para construção do ML, assegura a sua utilização para bases maiores. Além disso, as estratégias adotadas e apresentadas para adaptação da base de dados, incluindo o ML, ao *framework* Sphinx, podem servir testes futuros com outras bases.

5. Além de simplificações matemáticas nas equações utilizadas pelo algortimo de Viterbi,

foram incorporadas ao mesmo, as probabilidades fornecidas pelo Modelo Linguístico. Como foi exposto ao longo deste trabalho, modelos linguísticos (ML) capturam regularidades na linguagem falada e são utilizados no reconhecimento para estimar a probabilidade da sequência de palavras. Em um sistema de reconhecimento de palavras isoladas, é possível obter boas taxas de reconhecimento sem o uso de um ML, contudo este torna-se fundamental em sistemas de reconhecimento de fala contínua.

6. Diferentemente de vários trabalhos relacionados que são encontrados, os quais focam em uma ou duas etapas do processo de reconhecimento, este trabalho apresenta o modelo arquitetural de todas as etapas envolvidas. A arquitetura apresentada, além de contemplar todas as etapas, fornece detalhes quanto a: registradores, memórias R/W e ROM necessárias (incluindo as de controle), modelo das memórias, conteúdo das memórias calculas *offline*, tamanho dos barramentos, alguns *pseudo*códigos, entre outros.

7. Dado o fluxo de desenvolvimento de um sistema embarcado, apresentado no Capítulo 4, pode-se iniciar diretamente a etapa de construção do Modelo RTL. Embora esta não seja uma etapa trivial, a mesma consiste basicamente na codificação em uma linguagem de descrição de *hardware* (HDL). Uma vez que, todo o projeto arquitetural está definido, produto deste trabalho, a tarefa de codificação se torna bem mais simples e automática.

8. A decisão por realizar a etapa de treinamento em software e armazenar os modelos gerados em memórias que são lidas durante a etapa de decodificação, além de simplifcar a arquitetura do sistema, torna-o flexível a reconhecer novos padrões. Uma vez que, foram realizados testes com uma base de dados em português, incluindo o modelo linguístico, e foram obtidos boas taxas de reconhecimento, têm-se a garantia de que é possível utilizar os modelos gerados durante o treinamento no sistema embarcado a ser construído.

7.2 Sugestões para Trabalhos Futuros

Como foi exposto neste trabalho, o interesse por sistemas que realizem a interface homem-máquina de maneira mais natural e espontânea se torna iminente. Para isso, torna-se necessário a utilização de vocabulários cada vez maiores e que cubram possíveis variações de pronúncia. Assim, a seguir são apresentadas algumas sugestões para continuidade do trabalho apresentado.

1. Utilização de uma base de dados maior, de modo a ampliar o vocabulário do sistema. Essa tarefa se tornará mais fácil, uma vez que, neste trabalho foram apresentadas as estratégias adotadas para adaptar os principais arquivos necessários para realização de novos testes no *framework* Sphinx ou, em outra ferramenta semelhante;

2. Realização de experimentos objetivando a definição do limiar de corte para a etapa de cálculo das gaussianas, podendo os mesmos serem realizados em *software*;

3. Realização de experimentos para definir com maior precisão o tamanho em bits dos dados;

4. Implementação do Modelo RTL do sistema definido;

5. Ampliação do sistema para utilização do modelo *trigram*;

6. Investigação da técnica WFST que começa a ser utilizada em alguns dos trabalhos relacionados mais recentes.

Diante de todo o exposto, um impacto de grande relevância deste trabalho no âmbito científico consiste na adaptação das técnicas necessárias à realização de reconhecimento de fala contínua, projetados para a língua portuguesa, em sistemas com limitações de recursos de *hardware*. No âmbito tecnológico e econômico, a modelagem de um sistema que possibilite uma nova forma de comunicação entre o homem e dispositivos eletrônicos necessários ao seu cotidiano, é de interesse não só da comunidade consumidora, como também das empresas fabricantes de tais dispositivos. Além disso, a possibilidade de construção de equipamentos com essa capacidade, possui um impacto social junto, por exemplo, à comunidade de pessoas portadoras de deficiência física visual e/ou motora, que poderão dispor de um sistema de reconhecimento de fala, que possibilitará uma melhoria na sua comunicação com as máquinas. Sistemas fixos e treinados para reconhecer a fala de um usuário, para responder a sua solicitação, bem como reconhecer quem está falando, poderão ser facilmente instalados em repartições públicas, hospitais, postos policiais e empresas prestadoras de serviços.

Além dos impactos e repercussões citados, os resultados deste trabalho tornarão possível vislumbrar outras aplicações da comunicação vocal homem-máquina, visto que a máquina faz parte da vida do homem moderno e, portanto, a ciência deve buscar cada vez mais tornar mais fácil, atrativa e eficiente a interação do homem com as máquinas.

Apêndice A

Base de Dados

Listas de frases utilizadas neste trabalho.

LISTA 01

A questão foi retomada no congresso.
Leila tem um lindo jardim.
O analfabetismo é a vergonha do país.
A casa foi vendida sem pressa.
Trabalhando com união rende muito mais.
Recebi nosso amigo para almoçar.
A justiça é a única vencedora.
Isso se resolverá de forma tranquila.
Os pesquisadores acreditam nessa teoria.
Sei que atingiremos o objetivo.

LISTA 02

Nosso telefone quebrou.
Desculpe se magoei o velho.
Queremos discutir o orçamento.
Ela tem muita fome.
Uma índia andava na mata.
Zé, vá mais rápido! Hoje dormirei bem.
João deu pouco dinheiro.
Ainda são seis horas.
Ela saía discretamente.

LISTA 03

Eu vi logo a Iôiô e o Léo.
Um homem não caminha sem um fim.
Vi Zé fazer essas viagens seis vezes.
O atabaque do Tito é coberto com pele de gato.
Ele lê no leito de palha.
Paira um ar de arara rara no Rio Real.
Foi muito difícil entender a canção.
Depois do almoço te encontro.
Esses são nossos times.
Procurei Maria na copa.

LISTA 04

A pesca é proibida nesse lago.
Espero te achar bem quando voltar.
Temos muito orgulho da nossa gente.
O inspetor fez a vistoria completa.
Ainda não se sabe o dia da maratona.
Será muito difícil conseguir que eu venha.
A paixão dele é a natureza.
Você quer me dizer a data?
Desculpe, mas me atrasei no casamento.
Faz um desvio em direção ao mar!

LISTA 05

A velha leoa ainda aceita combater.
É hora do homem se humanizar mais.
Ela ficou na fazenda por uma hora.
Seu crime foi totalmente encoberto.
A escuridão da garagem assustou a criança.
Ontem não pude fazer minha ginástica.
Comer quindim é sempre uma boa pedida.
Hoje eu irei precisar de você.
Sem ele o tempo flui num ritmo suave.
A sujeira lançada no rio contamina os peixes.

LISTA 06

O jogo será transmitido bem tarde.
É possível que ele já esteja fora de perigo.
A explicação pode ser encontrada na tese.
Meu vôo tinha sido marcado para as cinco.
Daqui a pouco a gente irá pousar.
Estou certo que mereço a atenção dela.
Era um belo enfeite todo de palha.
O comércio daqui tem funcionado bem.
É a minha chance de esclarecer a notícia.
A visita transformou-se numa reunião íntima.

LISTA 07

O cenário da história é um subúrbio do Rio.
Eu tenho ótima razão para festejar.
A pequena nave medirá o campo magnético.
O prêmio será entregue sem sessão solene.
A ação se passa numa cidade calma.
Ela e o namorado vão a Portugal de navio.
O adiamento surpreendeu a mim e a todos.
A gente sempre colhe o que plantou.
Aqui é onde existem as flores mais interessantes.
A corrida de inverno aconteceu com vibração.

LISTA 08

Esse empreendimento será de enorme sucesso.
As feiras livres não funcionam amanhã.
Fumar é muito prejudicial à saúde.
Entre com seu código e o número da conta.
Reflita antes e discuta depois.
As aulas dele são bastante agradáveis.
Usar aditivos pode ser desastroso.
O clima não é mau em Calcutá.
A locomotiva vem sem muita carga.
Ainda é uma boa temporada para o cinema.

LISTA 09

Os maiores picos da Terra ficam debaixo d'água.
A inauguração da vila é quarta-feira.
Só vota quem tiver o título de eleitor.
É fundamental buscar a razão da existência.
A temperatura só é boa mais cedo.
Em muitas regiões a população está diminuindo.
Nunca se pode ficar em cima do muro.
Pra quem vê de fora o panorama é desolador.
É bom te ver colhendo flores.
Eu me banho no lago ao amanhecer.

LISTA 10

É fundamental chegar a uma solução comum.
Há previsão de muito nevoeiro no Rio.
Muitos móveis virão às cinco da tarde.
A casa pode desabar em algumas horas.
O candidato falou como se estivesse eleito.
A idéia é falha, mas interessa.
O dia está bom para passear no quintal.
Minhas correspondências não estão em casa.
A saída para a crise dele é o diálogo.
Finalmente o mau tempo deixou o continente.

LISTA 11

Um casal de gatos come no telhado.
A cantora foi apresentar seu último sucesso.
Lá é um lugar ótimo para tomar uns chopinhos.
O musical consumiu sete meses de ensaio.
Nosso baile inicia após as nove.
Apesar desses resultados, tomarei uma decisão.
A verdade não poupa nem as celebridades.
As queimadas devem diminuir este ano.
O vão entre o trem e a plataforma é muito grande.
Infelizmente não compareci ao encontro.

LISTA 12

As crianças conheceram o filhote de ema.
A bolsa de valores ficou em baixa.
O congresso volta atrás em sua palavra.
A médica receitou que eles mudassem de clima.
Não é permitido fumar no interior do ônibus.
A apresentação foi cancelada por causa do som.
Uma garota foi presa ontem à noite.
O prato do dia é couve com atum.
Eu viajarei ao Canadá amanhã.
A balsa é o meio de transporte daqui.

LISTA 13

O grêmio ganhou a quadra de esportes.
Hoje irei à vila sem meu filho.
Essa magia não acontece todo dia.
Será bom que você estude esse assunto.
O menu incluía pratos bem saborosos.
Podia dizer as horas, por favor?
A casa é ornamentada com flores do campo.
A Terra é farta, mas não infinita.
O sinal emitido é captado por receptores.
A mensalidade aumentou mais que a inflação.

LISTA 14

O tele-jornal termina às sete da noite.
A cabine telefônica fica na próxima rua.
Defender a ecologia é manter a vida.
Nesse verão o calor está insuportável.
Um jardim exige muito trabalho.
O mamão que eu comprei estava ótimo.
Meu primo falará com a gerência amanhã.
De dia apague a luz sempre.
A sociedade uruguaia tem que se mobilizar.
Suas atitudes são bem calmas.

LISTA 15

Dezenas de cabos eleitorais buscavam apoio.
A vitória foi paga com muito sangue.
Nossa filha tem amor por animais.
Esse peixe é mais fatal que certas cobras.
O time continua lutando pelo sucesso.
Essa medida foi devidamente alterada.
O estilete é uma arma perigosa.
Aguarde, quinta eu venho jantar em casa.
A mudança é lenta, porém duradoura.
O clima não é mais seco no interior.

LISTA 16

A sensibilidade indicará a escolha.
A Amazônia é a reserva ecológica do globo.
O ministério mudou demais com a eleição.
Novos rumos se abrem para a informática.
O capital de uma empresa depende da produção.
Se não fosse ela, tudo seria contido.
A principal personagem no filme é uma gueixa.
Receba seu jornal em sua casa.
A juventude tinha que revolucionar a escola.
A atriz terá quatro meses para ensaiar seu canto.

LISTA 17

Muito prazer em conhecê-lo.
Eles estavam sem um bom equipamento.
O sol ilumina a fachada de tarde.
A correção do exame está coerente.
As portas são antigas.
Sobrevoamos Natal acima das nuvens.
Trabalhei mais do que podia.
Hoje eu acordei muito calmo.
Esse canal é pouco informativo.
Parece que nascemos ontem.

LISTA 18

Receba meus parabéns pela apresentação.
Eu planejo uma viagem no feriado.
No lado de cá do rio há uma boa sombra.
A maioria dos visitantes gosta deste monumento.
Minha filha é especialista em música sacra.
A casa só tem um quarto.
A duração do simpósio é de cinco dias.
Ao contrário de nossa expectativa, correu tranquilo.
A intenção é obter apoio do governante.
A fila aumentou ao longo do dia.

LISTA 19

À noite a temperatura deve ir a zero.
A proposta foi inspecionada pela gerência.
O quadro mostra uma face do cotidiano.
Já era bem tarde quando ele me abordou.
O canário canta ao amanhecer.
A lojinha fica bem na esquina de casa.
Meu time se consagrou como o melhor.
Um instituto deve servir a sua meta.
Ele entende quando se fala pausadamente.
Seu saldo bancário está baixo.

LISTA 20

O termômetro marcava um grau.
O discurso de abertura é bem longo.
Eu precisei de microfone na conferência.
Joyce esticou sua temporada até quinta.
Nada como um almoço ao ar livre.
Nossa filha é a primeira aluna da classe.
Gostaria de deitar um pouco.
Não fizemos uma viagem muito cansativa.
Ainda tenho cinco telefonemas para dar.
O hotéis do sudoeste são fantásticos.

Resumo informativo dos locutores da base de dados.

Os locutores em destaque foram utilizados para os testes. Os demais, para o treinamento.

Locutores masculinos

locutor	listas	faixa etária	profissão	escolaridade	Cidade/Estado
m01	1 a 4	18 a 60 anos	estudante	superior	São Paulo - SP
m02	5 a 8	18 a 60 anos	engenheiro	superior	Piracicaba – SP
m03	13 a 16	18 a 60 anos	engenheiro	superior	Sta. B. D'Oeste - SP
m04	17 a 20	18 a 60 anos	engenheiro	superior	Fortaleza - CE
m05	9 a 12	18 a 60 anos	engenheiro	superior	S. Caetano do Sul - SP
m06	13 a 16	18 a 60 anos	analista sist.	superior	Ribeirão Preto - SP
m07	1 a 4	18 a 60 anos	professor	superior	São Carlos - SP
m09	9 a 12	18 a 60 anos	estudante	superior	Limeira - SP
m11	13 a 16	18 a 60 anos	estudante	superior	Tupã - SP
m12	17 a 20	18 a 60 anos	estudante	superior	São Paulo – SP
m13	17 a 20	18 a 60 anos	estudante	superior	Santos - SP
m14	1 a 4	18 a 60 anos	comerciante	2o grau	Cotia - SP
m15	17 a 20	18 a 60 anos	engenheiro	superior	Goiânia – GO
m16	5 a 8	18 a 60 anos	estudante	superior	Bauru – SP
m17	9 a 12	18 a 60 anos	estudante	superior	Porto Feliz - SP
m18	5 a 8	18 a 60 anos	estudante	superior	Brasília - DF
m20	1 a 4	18 a 60 anos	estudante	superior	São Paulo – SP
m21	9 a 12	18 a 60 anos	estudante	superior	São Paulo - SP
m23	5 a 8	18 a 60 anos	estudante	superior	Piracicaba - SP
m24	13 a 16	18 a 60 anos	estudante	superior	S. Joaquim da Barra - SP

Tabela A.1: *Locutores masculinos utilizados na base de dados.*

Locutores femininos

locutor	listas	faixa etária	profissão	escolaridade	Cidade/Estado
f01	17 a 20	18 a 60 anos	engenheira	superior	Barbacena - MG
f02	5 a 8	18 a 60 anos	bibliotecária	superior	São Carlos - SP
f03	9 a 12	18 a 60 anos	estudante	superior	S. Seb. Paraíso - MG
f04	13 a 16	18 a 60 anos	aux. serv. gerais	1o grau	Ribeirão Bonito - SP
f05	17 a 20	18 a 60 anos	analista sistemas	superior	São Carlos - SP
f06	1 a 4	18 a 60 anos	pedagoga	superior	Fortaleza - CE
f07	5 a 8	18 a 60 anos	secretária	superior	São Carlos - SP
f08	9 a 12	18 a 60 anos	secretária	superior	São Carlos - SP
f09	1 a 4	18 a 60 anos	comerciante	2o grau	São Carlos - SP
f10	13 a 16	18 a 60 anos	pedagoga	superior	São Paulo - SP
f11	17 a 20	18 a 60 anos	pedagoga	superior	Vitória - ES
f12	1 a 4	18 a 60 anos	fisioterapeuta	superior	Pindamonhangaba - SP
f13	1 a 4	18 a 60 anos	bióloga	superior	São Paulo - SP
f15	5 a 8	18 a 60 anos	balconista	2o grau	São Carlos - SP
f17	9 a 12	18 a 60 anos	música	superior	Santo André - SP
f18	9 a 12	18 a 60 anos	pedagoga	superior	Camocim S. Félix – PE
f19	13 a 16	18 a 60 anos	estudante	superior	Franca - SP
f20	5 a 8	18 a 60 anos	terap. ocupacional	superior	São Paulo - SP
f21	17 a 20	18 a 60 anos	estudante	superior	Jundiaí - SP
f22	13 a 16	+ 60 anos	aposentada	1o grau	Colatina - ES

Tabela A.2: *Locutores femininos utilizados na base de dados.*

Apêndice B

Valores dos Filtros Triangulares

Nas Tabelas B.1, B.2 e B.3 são apresentados os valores dos 36 filtros utilizados neste trabalho.

Tabela B.1: *Valores das 128 amostras dos filtros de 1 a 12.*

Amostra \ Filtro	**Filtro 1**	**Filtro 2**	**Filtro 3**	**Filtro 4**	**Filtro 5**	**Filtro 6**	**Filtro 7**	**Filtro 8**	**Filtro 9**	**Filtro 10**	**Filtro 11**	**Filtro 12**
1	0.2154	0	0	0	0	0	0	0	0	0	0	0
2	0.4307	0	0	0	0	0	0	0	0	0	0	0
3	0.6460	0	0	0	0	0	0	0	0	0	0	0
4	0.8614	0	0	0	0	0	0	0	0	0	0	0
5	0.7701	0.2299	0	0	0	0	0	0	0	0	0	0
6	0.1240	0.8760	0	0	0	0	0	0	0	0	0	0
7	0	0.4782	0.5218	0	0	0	0	0	0	0	0	0
8	0	0	0.8322	0.1678	0	0	0	0	0	0	0	0
9	0	0	0.1861	0.8139	0	0	0	0	0	0	0	0
10	0	0	0	0.5403	0.4597	0	0	0	0	0	0	0
11	0	0	0	0	0.8943	0.1057	0	0	0	0	0	0
12	0	0	0	0	0.2482	0.7518	0	0	0	0	0	0
13	0	0	0	0	0	0.6024	0.3976	0	0	0	0	0
14	0	0	0	0	0	0	0.9564	0.0436	0	0	0	0
15	0	0	0	0	0	0	0.3103	0.6897	0	0	0	0
16	0	0	0	0	0	0	0	0.6645	0.3355	0	0	0
17	0	0	0	0	0	0	0	0.0184	0.9816	0	0	0
18	0	0	0	0	0	0	0	0	0.3724	0.6276	0	0
19	0	0	0	0	0	0	0	0	0	0.7266	0.2734	0
20	0	0	0	0	0	0	0	0	0	0.0805	0.9195	0
21	0	0	0	0	0	0	0	0	0	0	0.4347	0.5653
22	0	0	0	0	0	0	0	0	0	0	0	0.7887
23	0	0	0	0	0	0	0	0	0	0	0	0.1426
24	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0

Continua na próxima página...

Tabela B.1 – Continuação

Filtro / Amostra	**Filtro 1**	**Filtro 2**	**Filtro 3**	**Filtro 4**	**Filtro 5**	**Filtro 6**	**Filtro 7**	**Filtro 8**	**Filtro 9**	**Filtro 10**	**Filtro 11**	**Filtro 12**
34	0	0	0	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0
38	0	0	0	0	0	0	0	0	0	0	0	0
39	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0	0	0
41	0	0	0	0	0	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0
43	0	0	0	0	0	0	0	0	0	0	0	0
44	0	0	0	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0	0	0	0
46	0	0	0	0	0	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	0	0	0	0
48	0	0	0	0	0	0	0	0	0	0	0	0
49	0	0	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0	0	0
51	0	0	0	0	0	0	0	0	0	0	0	0
52	0	0	0	0	0	0	0	0	0	0	0	0
53	0	0	0	0	0	0	0	0	0	0	0	0
54	0	0	0	0	0	0	0	0	0	0	0	0
55	0	0	0	0	0	0	0	0	0	0	0	0
56	0	0	0	0	0	0	0	0	0	0	0	0
57	0	0	0	0	0	0	0	0	0	0	0	0
58	0	0	0	0	0	0	0	0	0	0	0	0
59	0	0	0	0	0	0	0	0	0	0	0	0
60	0	0	0	0	0	0	0	0	0	0	0	0
61	0	0	0	0	0	0	0	0	0	0	0	0
62	0	0	0	0	0	0	0	0	0	0	0	0
63	0	0	0	0	0	0	0	0	0	0	0	0
64	0	0	0	0	0	0	0	0	0	0	0	0
65	0	0	0	0	0	0	0	0	0	0	0	0
66	0	0	0	0	0	0	0	0	0	0	0	0
67	0	0	0	0	0	0	0	0	0	0	0	0

Continua na próxima página...

Tabela B.1 – Continuação

Filtro / Amostra	**Filtro 1**	**Filtro 2**	**Filtro 3**	**Filtro 4**	**Filtro 5**	**Filtro 6**	**Filtro 7**	**Filtro 8**	**Filtro 9**	**Filtro 10**	**Filtro 11**	**Filtro 12**
68	0	0	0	0	0	0	0	0	0	0	0	0
69	0	0	0	0	0	0	0	0	0	0	0	0
70	0	0	0	0	0	0	0	0	0	0	0	0
71	0	0	0	0	0	0	0	0	0	0	0	0
72	0	0	0	0	0	0	0	0	0	0	0	0
73	0	0	0	0	0	0	0	0	0	0	0	0
74	0	0	0	0	0	0	0	0	0	0	0	0
75	0	0	0	0	0	0	0	0	0	0	0	0
76	0	0	0	0	0	0	0	0	0	0	0	0
77	0	0	0	0	0	0	0	0	0	0	0	0
78	0	0	0	0	0	0	0	0	0	0	0	0
79	0	0	0	0	0	0	0	0	0	0	0	0
80	0	0	0	0	0	0	0	0	0	0	0	0
81	0	0	0	0	0	0	0	0	0	0	0	0
82	0	0	0	0	0	0	0	0	0	0	0	0
83	0	0	0	0	0	0	0	0	0	0	0	0
84	0	0	0	0	0	0	0	0	0	0	0	0
85	0	0	0	0	0	0	0	0	0	0	0	0
86	0	0	0	0	0	0	0	0	0	0	0	0
87	0	0	0	0	0	0	0	0	0	0	0	0
88	0	0	0	0	0	0	0	0	0	0	0	0
89	0	0	0	0	0	0	0	0	0	0	0	0
90	0	0	0	0	0	0	0	0	0	0	0	0
91	0	0	0	0	0	0	0	0	0	0	0	0
92	0	0	0	0	0	0	0	0	0	0	0	0
93	0	0	0	0	0	0	0	0	0	0	0	0
94	0	0	0	0	0	0	0	0	0	0	0	0
95	0	0	0	0	0	0	0	0	0	0	0	0
96	0	0	0	0	0	0	0	0	0	0	0	0
97	0	0	0	0	0	0	0	0	0	0	0	0
98	0	0	0	0	0	0	0	0	0	0	0	0
99	0	0	0	0	0	0	0	0	0	0	0	0
100	0	0	0	0	0	0	0	0	0	0	0	0
101	0	0	0	0	0	0	0	0	0	0	0	0

Continua na próxima página...

Tabela B.1 – Continuação

Amostra \ Filtro	Filtro 1	Filtro 2	Filtro 3	Filtro 4	Filtro 5	Filtro 6	Filtro 7	Filtro 8	Filtro 9	Filtro 10	Filtro 11	Filtro 12
102	0	0	0	0	0	0	0	0	0	0	0	0
103	0	0	0	0	0	0	0	0	0	0	0	0
104	0	0	0	0	0	0	0	0	0	0	0	0
105	0	0	0	0	0	0	0	0	0	0	0	0
106	0	0	0	0	0	0	0	0	0	0	0	0
107	0	0	0	0	0	0	0	0	0	0	0	0
108	0	0	0	0	0	0	0	0	0	0	0	0
109	0	0	0	0	0	0	0	0	0	0	0	0
110	0	0	0	0	0	0	0	0	0	0	0	0
111	0	0	0	0	0	0	0	0	0	0	0	0
112	0	0	0	0	0	0	0	0	0	0	0	0
113	0	0	0	0	0	0	0	0	0	0	0	0
114	0	0	0	0	0	0	0	0	0	0	0	0
115	0	0	0	0	0	0	0	0	0	0	0	0
116	0	0	0	0	0	0	0	0	0	0	0	0
117	0	0	0	0	0	0	0	0	0	0	0	0
118	0	0	0	0	0	0	0	0	0	0	0	0
119	0	0	0	0	0	0	0	0	0	0	0	0
120	0	0	0	0	0	0	0	0	0	0	0	0
121	0	0	0	0	0	0	0	0	0	0	0	0
122	0	0	0	0	0	0	0	0	0	0	0	0
123	0	0	0	0	0	0	0	0	0	0	0	0
124	0	0	0	0	0	0	0	0	0	0	0	0
125	0	0	0	0	0	0	0	0	0	0	0	0
126	0	0	0	0	0	0	0	0	0	0	0	0
127	0	0	0	0	0	0	0	0	0	0	0	0
128	0	0	0	0	0	0	0	0	0	0	0	0

Tabela B.2: *Valores das 128 amostras dos filtros de 13 a 24.*

Amostra \ Filtro	Filtro 13	Filtro 14	Filtro 15	Filtro 16	Filtro 17	Filtro 18	Filtro 19	Filtro 20	Filtro 21	Filtro 22	Filtro 23	Filtro 24
1	0	0	0	0	0	0	0	0	0	0	0	0

Continua na próxima página...

Tabela B.2 – Continuação

Filtro / Amostra	**Filtro 13**	**Filtro 14**	**Filtro 15**	**Filtro 16**	**Filtro 17**	**Filtro 18**	**Filtro 19**	**Filtro 20**	**Filtro 21**	**Filtro 22**	**Filtro 23**	**Filtro 24**
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0
22	0.2113	0	0	0	0	0	0	0	0	0	0	0
23	0.8574	0	0	0	0	0	0	0	0	0	0	0
24	0.5297	0.4703	0	0	0	0	0	0	0	0	0	0
25	0	0.9308	0.0692	0	0	0	0	0	0	0	0	0
26	0	0.3666	0.6334	0	0	0	0	0	0	0	0	0
27	0	0	0.8154	0.1846	0	0	0	0	0	0	0	0
28	0	0	0.2880	0.7120	0	0	0	0	0	0	0	0
29	0	0	0	0.7763	0.2237	0	0	0	0	0	0	0
30	0	0	0	0.2836	0.7164	0	0	0	0	0	0	0
31	0	0	0	0	0.8044	0.1956	0	0	0	0	0	0
32	0	0	0	0	0.3438	0.6562	0	0	0	0	0	0
33	0	0	0	0	0	0.8910	0.1090	0	0	0	0	0
34	0	0	0	0	0	0.4605	0.5395	0	0	0	0	0
35	0	0	0	0	0	0.0301	0.9699	0	0	0	0	0

Continua na próxima página...

Tabela B.2 – Continuação

Filtro / Amostra	**Filtro 13**	**Filtro 14**	**Filtro 15**	**Filtro 16**	**Filtro 17**	**Filtro 18**	**Filtro 19**	**Filtro 20**	**Filtro 21**	**Filtro 22**	**Filtro 23**	**Filtro 24**
36	0	0	0	0	0	0	0.6258	0.3742	0	0	0	0
37	0	0	0	0	0	0	0.2235	0.7765	0	0	0	0
38	0	0	0	0	0	0	0	0.8330	0.1670	0	0	0
39	0	0	0	0	0	0	0	0.4570	0.5430	0	0	0
40	0	0	0	0	0	0	0	0.0810	0.9190	0	0	0
41	0	0	0	0	0	0	0	0	0.7244	0.2756	0	0
42	0	0	0	0	0	0	0	0	0.3730	0.6270	0	0
43	0	0	0	0	0	0	0	0	0.0216	0.9784	0	0
44	0	0	0	0	0	0	0	0	0	0.6919	0.3081	0
45	0	0	0	0	0	0	0	0	0	0.3635	0.6365	0
46	0	0	0	0	0	0	0	0	0	0.0352	0.9648	0
47	0	0	0	0	0	0	0	0	0	0	0.7259	0.2741
48	0	0	0	0	0	0	0	0	0	0	0.4190	0.5810
49	0	0	0	0	0	0	0	0	0	0	0.1122	0.8878
50	0	0	0	0	0	0	0	0	0	0	0	0.8180
51	0	0	0	0	0	0	0	0	0	0	0	0.5312
52	0	0	0	0	0	0	0	0	0	0	0	0.2444
53	0	0	0	0	0	0	0	0	0	0	0	0
54	0	0	0	0	0	0	0	0	0	0	0	0
55	0	0	0	0	0	0	0	0	0	0	0	0
56	0	0	0	0	0	0	0	0	0	0	0	0
57	0	0	0	0	0	0	0	0	0	0	0	0
58	0	0	0	0	0	0	0	0	0	0	0	0
59	0	0	0	0	0	0	0	0	0	0	0	0
60	0	0	0	0	0	0	0	0	0	0	0	0
61	0	0	0	0	0	0	0	0	0	0	0	0
62	0	0	0	0	0	0	0	0	0	0	0	0
63	0	0	0	0	0	0	0	0	0	0	0	0
64	0	0	0	0	0	0	0	0	0	0	0	0
65	0	0	0	0	0	0	0	0	0	0	0	0
66	0	0	0	0	0	0	0	0	0	0	0	0
67	0	0	0	0	0	0	0	0	0	0	0	0
68	0	0	0	0	0	0	0	0	0	0	0	0
69	0	0	0	0	0	0	0	0	0	0	0	0

Continua na próxima página...

Tabela B.2 – Continuação

Filtro / Amostra	**Filtro 13**	**Filtro 14**	**Filtro 15**	**Filtro 16**	**Filtro 17**	**Filtro 18**	**Filtro 19**	**Filtro 20**	**Filtro 21**	**Filtro 22**	**Filtro 23**	**Filtro 24**
70	0	0	0	0	0	0	0	0	0	0	0	0
71	0	0	0	0	0	0	0	0	0	0	0	0
72	0	0	0	0	0	0	0	0	0	0	0	0
73	0	0	0	0	0	0	0	0	0	0	0	0
74	0	0	0	0	0	0	0	0	0	0	0	0
75	0	0	0	0	0	0	0	0	0	0	0	0
76	0	0	0	0	0	0	0	0	0	0	0	0
77	0	0	0	0	0	0	0	0	0	0	0	0
78	0	0	0	0	0	0	0	0	0	0	0	0
79	0	0	0	0	0	0	0	0	0	0	0	0
80	0	0	0	0	0	0	0	0	0	0	0	0
81	0	0	0	0	0	0	0	0	0	0	0	0
82	0	0	0	0	0	0	0	0	0	0	0	0
83	0	0	0	0	0	0	0	0	0	0	0	0
84	0	0	0	0	0	0	0	0	0	0	0	0
85	0	0	0	0	0	0	0	0	0	0	0	0
86	0	0	0	0	0	0	0	0	0	0	0	0
87	0	0	0	0	0	0	0	0	0	0	0	0
88	0	0	0	0	0	0	0	0	0	0	0	0
89	0	0	0	0	0	0	0	0	0	0	0	0
90	0	0	0	0	0	0	0	0	0	0	0	0
91	0	0	0	0	0	0	0	0	0	0	0	0
92	0	0	0	0	0	0	0	0	0	0	0	0
93	0	0	0	0	0	0	0	0	0	0	0	0
94	0	0	0	0	0	0	0	0	0	0	0	0
95	0	0	0	0	0	0	0	0	0	0	0	0
96	0	0	0	0	0	0	0	0	0	0	0	0
97	0	0	0	0	0	0	0	0	0	0	0	0
98	0	0	0	0	0	0	0	0	0	0	0	0
99	0	0	0	0	0	0	0	0	0	0	0	0
100	0	0	0	0	0	0	0	0	0	0	0	0
101	0	0	0	0	0	0	0	0	0	0	0	0
102	0	0	0	0	0	0	0	0	0	0	0	0
103	0	0	0	0	0	0	0	0	0	0	0	0

Continua na próxima página...

Tabela B.2 – Continuação

Amostra \ Filtro	Filtro 13	Filtro 14	Filtro 15	Filtro 16	Filtro 17	Filtro 18	Filtro 19	Filtro 20	Filtro 21	Filtro 22	Filtro 23	Filtro 24
104	0	0	0	0	0	0	0	0	0	0	0	0
105	0	0	0	0	0	0	0	0	0	0	0	0
106	0	0	0	0	0	0	0	0	0	0	0	0
107	0	0	0	0	0	0	0	0	0	0	0	0
108	0	0	0	0	0	0	0	0	0	0	0	0
109	0	0	0	0	0	0	0	0	0	0	0	0
110	0	0	0	0	0	0	0	0	0	0	0	0
111	0	0	0	0	0	0	0	0	0	0	0	0
112	0	0	0	0	0	0	0	0	0	0	0	0
113	0	0	0	0	0	0	0	0	0	0	0	0
114	0	0	0	0	0	0	0	0	0	0	0	0
115	0	0	0	0	0	0	0	0	0	0	0	0
116	0	0	0	0	0	0	0	0	0	0	0	0
117	0	0	0	0	0	0	0	0	0	0	0	0
118	0	0	0	0	0	0	0	0	0	0	0	0
119	0	0	0	0	0	0	0	0	0	0	0	0
120	0	0	0	0	0	0	0	0	0	0	0	0
121	0	0	0	0	0	0	0	0	0	0	0	0
122	0	0	0	0	0	0	0	0	0	0	0	0
123	0	0	0	0	0	0	0	0	0	0	0	0
124	0	0	0	0	0	0	0	0	0	0	0	0
125	0	0	0	0	0	0	0	0	0	0	0	0
126	0	0	0	0	0	0	0	0	0	0	0	0
127	0	0	0	0	0	0	0	0	0	0	0	0
128	0	0	0	0	0	0	0	0	0	0	0	0

Tabela B.3: *Valores das 128 amostras dos filtros de 25 a 36.*

Amostra \ Filtro	Filtro 25	Filtro 26	Filtro 27	Filtro 28	Filtro 29	Filtro 30	Filtro 31	Filtro 32	Filtro 33	Filtro 34	Filtro 35	Filtro 36
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0

Continua na próxima página...

Tabela B.3 – Continuação

Filtro / Amostra	**Filtro 25**	**Filtro 26**	**Filtro 27**	**Filtro 28**	**Filtro 29**	**Filtro 30**	**Filtro 31**	**Filtro 32**	**Filtro 33**	**Filtro 34**	**Filtro 35**	**Filtro 36**
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0

Continua na próxima página...

Tabela B.3 – Continuação

Filtro / Amostra	Filtro 25	Filtro 26	Filtro 27	Filtro 28	Filtro 29	Filtro 30	Filtro 31	Filtro 32	Filtro 33	Filtro 34	Filtro 35	Filtro 36
38	0	0	0	0	0	0	0	0	0	0	0	0
39	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0	0	0
41	0	0	0	0	0	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0
43	0	0	0	0	0	0	0	0	0	0	0	0
44	0	0	0	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0	0	0	0
46	0	0	0	0	0	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	0	0	0	0
48	0	0	0	0	0	0	0	0	0	0	0	0
49	0	0	0	0	0	0	0	0	0	0	0	0
50	0.1820	0	0	0	0	0	0	0	0	0	0	0
51	0.4688	0	0	0	0	0	0	0	0	0	0	0
52	0.7556	0	0	0	0	0	0	0	0	0	0	0
53	0.9604	0.0396	0	0	0	0	0	0	0	0	0	0
54	0.6923	0.3077	0	0	0	0	0	0	0	0	0	0
55	0.4242	0.5758	0	0	0	0	0	0	0	0	0	0
56	0.1562	0.8438	0	0	0	0	0	0	0	0	0	0
57	0	0.8954	0.1046	0	0	0	0	0	0	0	0	0
58	0	0.6449	0.3551	0	0	0	0	0	0	0	0	0
59	0	0.3944	0.6056	0	0	0	0	0	0	0	0	0
60	0	0.1439	0.8561	0	0	0	0	0	0	0	0	0
61	0	0	0.9004	0.0996	0	0	0	0	0	0	0	0
62	0	0	0.6662	0.3338	0	0	0	0	0	0	0	0
63	0	0	0.4321	0.5679	0	0	0	0	0	0	0	0
64	0	0	0.1980	0.8020	0	0	0	0	0	0	0	0
65	0	0	0	0.9662	0.0338	0	0	0	0	0	0	0
66	0	0	0	0.7474	0.2526	0	0	0	0	0	0	0
67	0	0	0	0.5286	0.4714	0	0	0	0	0	0	0
68	0	0	0	0.3098	0.6902	0	0	0	0	0	0	0
69	0	0	0	0.0910	0.9090	0	0	0	0	0	0	0
70	0	0	0	0	0.8805	0.1195	0	0	0	0	0	0
71	0	0	0	0	0.6761	0.3239	0	0	0	0	0	0

Continua na próxima página...

Tabela B.3 – Continuação

Filtro / Amostra	**Filtro 25**	**Filtro 26**	**Filtro 27**	**Filtro 28**	**Filtro 29**	**Filtro 30**	**Filtro 31**	**Filtro 32**	**Filtro 33**	**Filtro 34**	**Filtro 35**	**Filtro 36**
72	0	0	0	0	0.4716	0.5284	0	0	0	0	0	0
73	0	0	0	0	0.2670	0.7330	0	0	0	0	0	0
74	0	0	0	0	0.0626	0.9374	0	0	0	0	0	0
75	0	0	0	0	0	0.8674	0.1326	0	0	0	0	0
76	0	0	0	0	0	0.6762	0.3238	0	0	0	0	0
77	0	0	0	0	0	0.4851	0.5149	0	0	0	0	0
78	0	0	0	0	0	0.2940	0.7060	0	0	0	0	0
79	0	0	0	0	0	0.1029	0.8971	0	0	0	0	0
80	0	0	0	0	0	0	0.9175	0.0825	0	0	0	0
81	0	0	0	0	0	0	0.7389	0.2611	0	0	0	0
82	0	0	0	0	0	0	0.5603	0.4397	0	0	0	0
83	0	0	0	0	0	0	0.3817	0.6183	0	0	0	0
84	0	0	0	0	0	0	0.2031	0.7969	0	0	0	0
85	0	0	0	0	0	0	0.0245	0.9755	0	0	0	0
86	0	0	0	0	0	0	0	0.8559	0.1441	0	0	0
87	0	0	0	0	0	0	0	0.6890	0.3110	0	0	0
88	0	0	0	0	0	0	0	0.5221	0.4779	0	0	0
89	0	0	0	0	0	0	0	0.3551	0.6449	0	0	0
90	0	0	0	0	0	0	0	0.1882	0.8118	0	0	0
91	0	0	0	0	0	0	0	0.0213	0.9787	0	0	0
92	0	0	0	0	0	0	0	0	0.8639	0.1361	0	0
93	0	0	0	0	0	0	0	0	0.7078	0.2922	0	0
94	0	0	0	0	0	0	0	0	0.5519	0.4481	0	0
95	0	0	0	0	0	0	0	0	0.3958	0.6042	0	0
96	0	0	0	0	0	0	0	0	0.2398	0.7602	0	0
97	0	0	0	0	0	0	0	0	0.0838	0.9162	0	0
98	0	0	0	0	0	0	0	0	0	0.9325	0.0675	0
99	0	0	0	0	0	0	0	0	0	0.7867	0.2133	0
100	0	0	0	0	0	0	0	0	0	0.6409	0.3591	0
101	0	0	0	0	0	0	0	0	0	0.4951	0.5049	0
102	0	0	0	0	0	0	0	0	0	0.3493	0.6507	0
103	0	0	0	0	0	0	0	0	0	0.2035	0.7965	0
104	0	0	0	0	0	0	0	0	0	0.0577	0.9423	0
105	0	0	0	0	0	0	0	0	0	0	0.9177	0.0823

Continua na próxima página...

Tabela B.3 – Continuação

Filtro / Amostra	**Filtro 25**	**Filtro 26**	**Filtro 27**	**Filtro 28**	**Filtro 29**	**Filtro 30**	**Filtro 31**	**Filtro 32**	**Filtro 33**	**Filtro 34**	**Filtro 35**	**Filtro 36**
106	0	0	0	0	0	0	0	0	0	0	0.7814	0.2186
107	0	0	0	0	0	0	0	0	0	0	0.6452	0.3548
108	0	0	0	0	0	0	0	0	0	0	0.5089	0.4911
109	0	0	0	0	0	0	0	0	0	0	0.3726	0.6274
110	0	0	0	0	0	0	0	0	0	0	0.2364	0.7636
111	0	0	0	0	0	0	0	0	0	0	0.1001	0.8999
112	0	0	0	0	0	0	0	0	0	0	0	0.9662
113	0	0	0	0	0	0	0	0	0	0	0	0.8389
114	0	0	0	0	0	0	0	0	0	0	0	0.7115
115	0	0	0	0	0	0	0	0	0	0	0	0.5841
116	0	0	0	0	0	0	0	0	0	0	0	0.4568
117	0	0	0	0	0	0	0	0	0	0	0	0.3294
118	0	0	0	0	0	0	0	0	0	0	0	0.2021
119	0	0	0	0	0	0	0	0	0	0	0	0.0748
120	0	0	0	0	0	0	0	0	0	0	0	0
121	0	0	0	0	0	0	0	0	0	0	0	0
122	0	0	0	0	0	0	0	0	0	0	0	0
123	0	0	0	0	0	0	0	0	0	0	0	0
124	0	0	0	0	0	0	0	0	0	0	0	0
125	0	0	0	0	0	0	0	0	0	0	0	0
126	0	0	0	0	0	0	0	0	0	0	0	0
127	0	0	0	0	0	0	0	0	0	0	0	0
128	0	0	0	0	0	0	0	0	0	0	0	0

Apêndice C

Publicações

Neste apêndice são apresentadas as publicações relacionadas a este trabalho.

Evaluation of the impact in reducing the number of parameters for continuous speech recognition for Brazilian Portuguese

Daniella Dias Cavalcante da Silva[1]
and César Rocha Vasconcelos[2]
Telematics Coordination
Federal Institute of Technological Education - IFPB
Campina Grande, Brazil
Emails: {daniella.silva[1], cesarocha[2]}@ifpb.edu.br

Benedito Guimarães Aguiar Neto[3]
and Joseana Macêdo Fechine[4]
Computer and Electrical Engineering Center
Federal University of Campina Grande - UFCG
Campina Grande, Brazil
Emails: bganeto@dee.ufcg.edu.br[3],
joseana@dsc.ufcg.edu.br[4]

***Abstract*—With the advent of technology, machines predominate in almost all scenarios of everyday life. The possibility of performing human-machine communication through speech makes this interaction easier and more productive. However, processing requirements still make difficult the implementation of systems for automatic speech recognition on devices with low computational power such as mobile phones, palmtops and appliances. For this purpose, it becomes necessary to research for optimized acoustic and language modeling, associated with use of representative databases, looking for a good compromise between recognition rates and computational demands imposed by embedded systems. The main goal of this work is to model the architecture of a system for continuous speech recognition to Brazilian Portuguese, in order to enable its implementation in an embedded system with limited computing resources.**

***Index Terms*—Embedded System, Hidden Markov Models, MFCC, Speech Recognition.**

I. Introduction

It is widely recognized that, with the growth of technology, machines such as computers, appliances and portable consuming devices have played an important role in almost all scenarios of everyday life. In this sense, the human-machine communication by voice has enabled people to interact with machines in an easier and even more productive way. Some researchers argued that this form of interaction is also more natural to humans because it allows hands and users'eyes available for performing other tasks.

Prominent speech recognition approaches such as Hidden Markov Models (HMM) have been using statistics models that can learn by examples and also consider other factors, like speech variations and noising environments during the speech recognition process [1, 2, 3, 4, 5]. The effect of variables not modeled such as differences in channel, microphone types, lack of vocabulary or sub-phonetic units trained poorly, can seriously compromise the performance and accuracy of speech recognition systems.

It's certainly true that different languages have different phonemes, both in number and characteristics (duration, intensity, etc.). This means that parameters and settings used in speech recognition systems that provide high recognition rates in a given language may not be suitable for another one.

It has been discussed that processing limitations inherent in small computing power devices like cell phones, palmtops and appliances have hindered implementation of automatic speech recognition systems for these kind of devices. Thus, it is crucial to research for optimized acoustic and language modeling associated with the use of representative databases that may demonstrate a good compromise between recognition rates and computational characteristics imposed by embedded systems.

Clearly, for a efficient and generic system speech recognition, it becomes necessary to adjust the acoustic models to the Brazilian Portuguese language [1, 3, 6, 7]. Some researchers have dedicated efforts for obtaining efficient methods for modeling and processing speech in Portuguese language [1, 3, 4, 5]. However, the research for a more detailed and representative modeling of brazilian portuguese language well-suited for small computing power devices is still a promising area.

In this work, an architecture of a continuous speech recognition system suited to Brazilian Portuguese language, using HMM, is presented to enable its implementation in an embedded system with limited computing resources. Also, the results of experiments performed using the Sphinx framework [8] analyzing the speech recognition rates and computational costs are discussed.

This paper is organized as follows. Section II briefly describes the database used in this work. In Section III the development methodology is explained. In section IV the experiment scenarios are described. Section V shows the experimental results followed by their analysis in Section VI. The concluding remarks are made in Section VII.

II. Database and Language Model

Availability of databases is determinant for developing of continuous speech recognition systems. Sharing of databases between U.S. and European researchers have been a major reason for the progress obtained in their respective countries over the past decades. However, to provide sufficient examples for statistical methods work properly, the database needs to be representative and therefore very hard to build, both in time spent and financial terms [6, 9]. For Brazilian Portuguese language there is still a great lack of shared databases, so that only individual efforts of research centers are eventually found [10, 11, 12].

In this paper it was used the database developed in [10]. This choice was made because its structure is suitable for continuous speech recognition applications. Database is composed of 46 different speakers, being 24 males and 22 females of different ages (18 to 60 years), degrees of instruction and cities in Brazil. In addition, 200 phonetically balanced sentences were used [9], which ensures the representation of all phonemes of Brazilian Portuguese language. Last but not least, another reason that contributed to this choice was database availability.

Originally, the database was composed of:

- 1600 audio files in WAV format with a sampling frequency of 11kHz and 16bit resolution;
- Phonetic transcription files;
- Phonetic dictionary (phonemes);
- Dictionary (vocabulary words).

Another key element in achieving continuous speech recognition is the Language Model (LM). The original database did not include an LM, which had to be built. For this task, it was used the *lmtool*, which is available at http://www.speech.cs.cmu.edu/tools/lmtool-new.html. LM is built using the N-gram model, based on the sentences corpus. The file generated by the tool lmtool is in text format. However, Sphinx uses the standard ARPA, whose format is DMP. To perform this conversion, the tool *sphinx_lm_convert* is available.

III. Methodology

The main goal of this work is to find the best compromise between recognition rate and limitations of an embedded hardware. So, the first step was to define the acoustic and language modeling to be adopted, followed by evaluation of parameters used. The methodology for performing this step is presented below.

Two methods of spectral analysis predominate in speech recognition: *Linear Predictive Coding*) [13, 14, 15] and *Spectral Analysis Filter Bank* [16, 17, 18]. Researches have shown that for the purpose of continuous speech recognition, the best recognition rates are obtained by using mel frequency cepstral coefficients (MFCC) [4, 5, 7, 16, 17, 18, 19]. Process of obtaining MFCC is illustrated in Fig. 1.

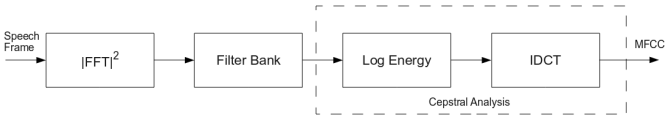


Fig. 1. Process of obtaining MFCC.

Fig. 2. Features vector.

First, the module of the FFT of samples belonging to the analysis window is calculated. Next, the magnitude frequency response is multiplied by a set of triangular bandpass filters. Then, the logarithm of the energy of the output filters is calculated. Finally, the IDCT (Inverse Discrete Cosine Transform) is applied in these values. Thereafter, mel frequency cepstral coefficients are obtained. This process can be represented mathematically by [20]:

$$c_{mel}(n) = \sum_{k=1}^{M} log(S(k)) \cdot cos[n(k - \frac{1}{2})] \cdot \frac{\pi}{M} \quad , n = 0, 1, \ldots, M \tag{1}$$

where M is the number of filters used, $c_{mel}(n)$ is the coefficient of index n and $S(k)$ the output signal of filters bank, which is defined by:

$$S(k) = \sum_{j=1}^{NFFT} W_k(j) \cdot X(j) \quad , k = 1, \ldots, M, \tag{2}$$

where $W_k(j)$ are the triangular weighting windows spaced along the Mel frequency and $X(j)$ is the spectrum of the magnitude of FTT of N points [20].

Typically, delta-cepstral and double-delta cepstral coefficients are appended to MFCC features in order to add dynamic information to the static cepstral features. Thus, the feature vector is now composed by N mel frequency cepstral coefficients with their delta and double-delta cepstral, obtaining a total of $3 * N$ elements (Fig. 2).

Calculation of delta based on MFCC [21] is given by:

$$d(n) \equiv \frac{d}{dt} c(n) \approx c(n+2) - c(n-2) \tag{3}$$

Double-delta parameters are obtained by reapplying the derivative on the results hold in the first derivation [21]:

$$d2(n) \equiv \frac{d}{dt} d(n) \approx d(n+2) - d(n-2) \tag{4}$$

The process of obtaining delta and double-delta from MFCC is illustrated graphically in Fig. 3. Delta is computed by subtracting the cepstrum that is two frames behind of the current cepstrum from the cepstrum that is two

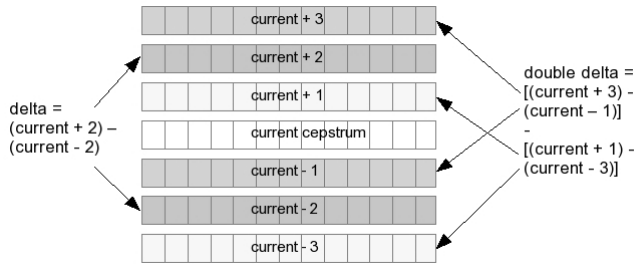


Fig. 3. Process of obtaining delta and double-delta from MFCC.

frames ahead of the current cepstrum. The computation of the double delta is similar. It is computed by subtracting the delta cepstrum one time frame behind from the delta cepstrum one time frame ahead.

Experiments are performed in framework Sphinx using MFCC with their first and second derivatives, varying the number of filters and coefficients in order to find the best configuration to be used in a embedded context.

In this work, Hidden Markov Models are used to build accoustic model and classification. It is important to notice that several parameters can be varied in this model, such as: type of HMM (discrete, continuous or semi-continuous) and number of states. State of the art shows that for continuous speech recognition the best results are obtained with continuous HMM. So, experiments are performed to identify more appropriate number of states to model Brazilian Portuguese.

Language Models (LM) capture the regularities in the spoken language and is used by the speech recognizer to estimate the probability of word sequences. In a system for isolated words recognition it is possible obtain good results without a Language Model. However, this is fundamental in a continuous speech recognition system. There are many techniques to build these models, such as: N-gram, decision trees and context-free grammar. Recent researches indicate that N-gram is one of the models that offer better results [6, 22, 23, 24, 25]. So, it was the language model chosen in this work.

As mentioned before, so many parameters can be varied in the configuration of a speech recognition system using MFCC, HMM and N-gram. The configuration of these parameters can increase or decrease the recognition rate against more or less computational cost. In this work, it was investigated the impact in recognition accuracy by reducing the following parameters:

- Number of HMM states (3 and 5);
- Number of filters for MFCC calculation(20 to 40); '
- Number of MFCC (8 to 10);

To obtain an estimate of the computational cost impact for each configuration, it was measured the time of execution for the following stages of the system:

- Features extraction in training;
- Training;
- Features extraction in decoding;
- Decoding.

For each experimented configuration, it was necessary a evaluation of the recognition rate against time of execution. This aims to identify the better configurations. Among these candidate configurations, one will be used in architecture of the system intended in this work. Results obtained in this stage are presented in section V and decisions adopted based on the results in section VI.

IV. Experiment Scenarios

Experiments were realized using framework Sphinx3 in a machine with AMD Turion X2 processor of 800MHz and 2GB of principal memory RAM. As mentioned in Section II, the database used is composed of 200 phonetically balanced sentences, pronounced by 46 different speakers. The sentences that compose database and information about the speakers can be found at [9].

Experiment scenarios are described below:

- 1200 sentences for training and 400 for decoding;
- 19 male speakers and 17 female speakers for training;
- 05 speakers of each gender, a total of 10 speakers for decoding;
- HMM with 3 and 5 states;
- 8 to 10 mel frequency cepstral coefficients (MFCC);
- mel-scale filter bank using from 20 to 40 filters;

Speakers from decoding and training stages are different. This ensures the independence from speakers in recognition process.

Since the goal of this study is to identify a setting that gives good recognition rates without requiring many computational resources, experiments were performed with values smaller than the ones that literature suggests: 12 or 13 MFCC [4, 5, 7, 19]. The purpose of these experiments was to evaluate the impact of using a small number of coefficients in the recognition rate and in the execution time for training and decoding stages. It is worth noting, that time depends on the configuration of the machine on which the simulations were performed. More accurate metrics could be used, such as: number of clock cycles or number of arithmetic and logic operations required. However, since all simulations were performed in the same scenario, i.e. on the same machine, it is possible to have a normalized view of computational overhead for executing each step. More accurate metrics, as mentioned, will be used in the next step of this work through a profiling tool.

V. Results

The graphs presented in this section will enable analysis of the relationships between:

- Number of coefficients and recognition rate;
- Number of coefficients and training and decoding time;
- Number of filters and recognition rate;
- Number of filters and training and decoding time.

The rate of sentence recognition based on the number of filters used to calculate MFCC in 3-state HMM is shown in Fig. 4.

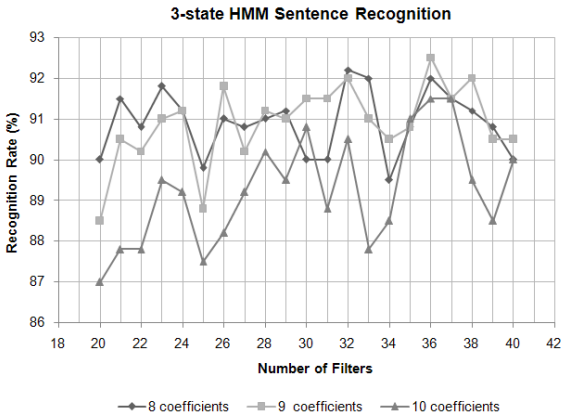


Fig. 4. Relationship between sentence recognition rate and number of filters for 3-state HMM, depending on the number of coefficients.

The rate of sentence recognition based on the number of filters used to calculate MFCC in 5-state HMM is shown in Fig. 5.

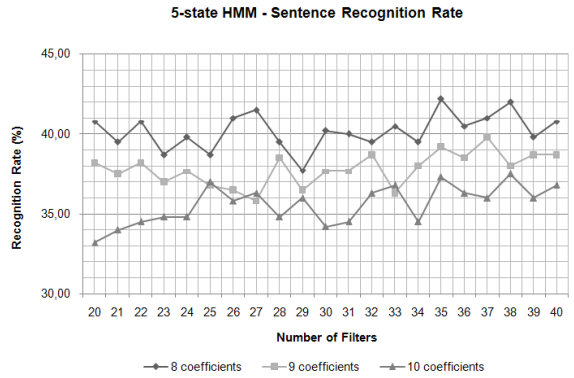


Fig. 5. Relationship between sentence recognition rate and number of filters for 5-state HMM, depending on the number of coefficients.

The training time was measured against the variation of the filters and coefficients numbers. It was observed that the number of filters almost did not affect the training and recognition time but rather the number of coefficients. The relationship between the time required for recognition of 400 sentences and the number of filters with 3-state HMM is shown in Fig. 6.

The average training and recognition time based on the number of coefficients with 3-state HMM is shown in Fig. 7.

It is important to notice that in charts were informed recognition rates for complete sentences and not for isolated words. In some cases, a sentence can not be fully recognized, but only some of its words. For example, in the sentence "Muito prazer em conhecê -lo", which is composed

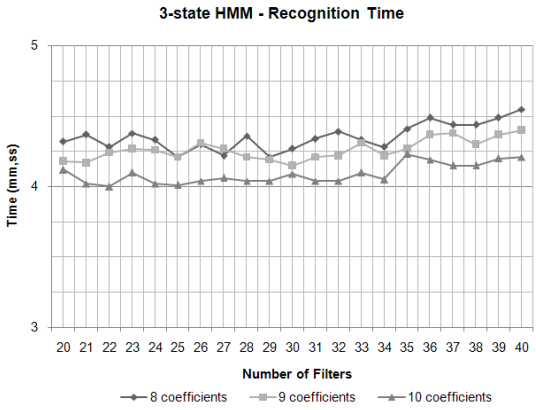


Fig. 6. Relationship between recognition time and number of filters for 3-state HMM based on the number of coefficients.

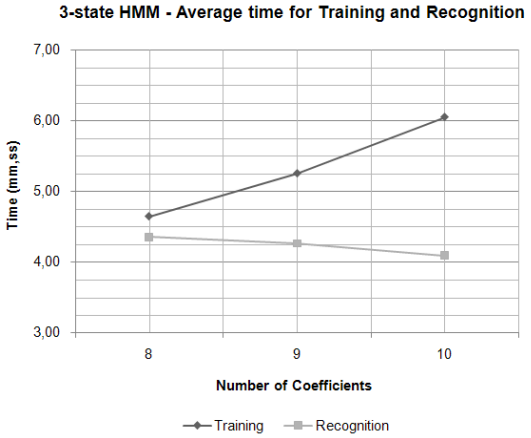


Fig. 7. Relationship between training and recognition time and the number of coefficients for 3-state HMM.

of five words, if only "lo" was not recognized, this would yield a recognition rate of 80% for words and 0% for sentences. Despite a rate of 0% in sentence recognition, since 80% of words were recognized, it is possible that with some adjustments in the parameters, the sentence would be fully recognized. However, as this work aims to perform continuous speech recognition, the charts that depicts such results have been omitted in this document.

VI. Results Analysis

Based on presented results it is possible observe that accuracy rates over 92% has being obtained and the best ones with 3-state HMM, 9 coefficients and 36 filters. Another important information refers to the time required for training and decoding by varying the number of coefficients. Considering the results, the following decisions were made for the architecture of the recognition system to be modeled:

1) Number of HMM states

   As reported previously, the best recognition rates were obtained with 3-state HMM. So this is the

number of states to be adopted in this work. Although a larger number of states should provide a higher recognition rate, it is important to highlight that the model was trained to model phonemes and the property *skipstate* is not allowed. This can force the HMM to go through more states than it is necessary to represent certain phoneme, thus increasing the error.

2) Number of coefficients

Considering the results presented, it is possible to observe that the number of coefficients directly influences the training time. The lower the number of coefficients, less training time. However, the time required for recognition increases. Taking into account the results of experiments and the mentioned aspects, it was decided to use 9 coefficients.

3) Number of filters

For sentences recognition the best rates were obtained with 36 filters. As the number of filters did not influence significantly the time of recognition, the option to use this number of filters becomes a strong candidate to be used in modeling the architecture to be implemented. However, it is necessary to evaluate this decision before more accurate metrics such as those cited earlier (e.g. number of clock cycles, arithmetic and logic operations). This will be accomplished in the next stages of work.

4) Use of an external memory for models storage

Analyzing the time needed for training models and considering some related work [26, 27, 28, 29, 30], this step is performed in a offline mode in software, since it requires more processing power. The acoustic and language models will be stored in an external memory being used at the time of recognition. This strategy also becomes interesting because it is possible to adapt the system to new patterns, requiring only that memory is updated with new acoustic and language models.

VII. Final Considerations

In this paper, the results of experiments performed to identify the best configuration for automatic continuous speech recognition suited to Brazilian Portuguese were presented. During these experiments, it was evaluated the number of states of the HMM, the number of mel-frequency cepstral coefficients and number of filters to obtain the coefficients.

The results show that is possible to establish settings in order to identify a suitable configuration to be used for speech recognition at an specific embedded context with limited computing resources. Future work will focus on employing a profiling tool in order to measure the computational cost of the candidate configurations. Such configurations will be evaluated in terms of the number of clock cycles and memory accesses. After that, it will be possible to select the best configuration combined with the lowest cost.

Furthermore, the system performance can be experimented with other approaches. Other parameters and settings on the Sphinx framework will be investigated to find out a representative modeling that may improve speech recognition rates on embedded systems with limited computing resources.

References

[1] V. F. S. de Alencar and A. Alcaim, "Lsf and lpc - derived features for large vocabulary distributed continuous speech recognition in brazilian portuguese," 2008.

[2] A. A. BRESOLIN, A. D. D. NETO, and P. J. ALSINA, "Brazilian vowels recognition using a new hierarchical decision structure with wavelet packet and svm," 2007.

[3] A. A. BRESOLIN, A. D. D. NETO, and P. J. ALSINA, "Digit recognition using wavelet and svm in brazilian portuguese," 2008.

[4] P. Silva, N. Neto, A. Klautau, A. Adami, and I. Trancoso, "Speech recognition for brazilian portuguese using the spoltech and ogi-22 corpora," 2008.

[5] C. HOSN, L. A. BAPTISTA, T. IMBIRIBIRA, and A. KLAUTAU, "New resources for brazilian portuguese: results for grapheme-to-phoneme and phone classification," 2006.

[6] R. T. Tevah, *Implementação de um sistema de reconhecimento de fala contínua com amplo vocabulário para o português brasileiro.* PhD thesis, 2006.

[7] G. F. G. Yared and F. Violaro, "Algoritmo para redução do número de parâmetros de modelos hmm utilizados em sistemas de reconhecimento de fala contínua," 2005.

[8] K. LEE, H. HON, and R. REDDY, "An overview of the sphinx speech recognition system," 1990.

[9] A. Alcaim, J. A. SOLEWICZ, and J. A. MORAES, "Freqüência de ocorrência dos fones e listas de frases foneticamente balanceadas no português falado no rio de janeiro," *Revista da Sociedade Brasileira de Telecomunicações (SBrT)*, vol. 7, no. 1, pp. 23–41, 1992.

[10] C. A. Ynoguti and F. Violaro, "A brazilian portuguese speech database," 2008.

[11] "Spoltech brazilian portuguese corpus," 2010.

[12] "Corpus de extractus de textos electrônicos nilc/ folha de são paulo (ceten-folha)," 2010.

[13] L. Rabiner and R. W. Schafer, *Digital Processing of*

Speech Signals. Englewood Cliffs, NJ: Prentice-Hall, 1978.

[14] N. Zheng and P. Ching, "Using haar transformed vocal source information for automatic speaker recognition," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, vol. 1, IEEE, 2004.

[15] B. Atal, "The history of linear prediction," *IEEE Signal Processing Magazine*, vol. 23, no. 2, pp. 154–161, 2006.

[16] G. Garau and S. Renals, "Combining spectral representations for large-vocabulary continuous speech recognition," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 16, no. 3, pp. 508–518, 2008.

[17] R. Dias, "Normalização de locutor em sistema de reconhecimento de fala," Master's thesis, Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, Brasil, 2000.

[18] C. Lee, D. Hyun, E. Choi, J. Go, and C. Lee, "Optimizing feature extraction for speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 1, pp. 80–87, 2003.

[19] E. SILVA, L. BAPTISTA, H. FERNANDES, and A. KLAUTAU, "Desenvolvimento de um sistema de reconhecimento automático de voz contínua com grande vocabulário para o português brasileiro," 2005.

[20] J. Godino-Llorente, P. Gomez-Vilda, and M. Blanco-Velasco, "Dimensionality reduction of a pathological voice quality assessment system based on Gaussian mixture models and short-term cepstral parameters," *Biomedical Engineering, IEEE Transactions on*, vol. 53, no. 10, pp. 1943–1953, 2006.

[21] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*. Prentice-Hall, 1993.

[22] Y.-Y. Wang, D. Yu, Y.-C. Ju, and A. Acero, "An introduction to voice search," vol. 25, no. 3, pp. 28–38, 2008.

[23] P. Fung and T. Schultz, "Multilingual spoken language processing," vol. 25, no. 3, pp. 89–97, 2008.

[24] M. RAJMAN, *Speech and Language Engineering*. EPFL Press, 2007.

[25] D. Yu, Y.-C. Ju, Y.-Y. Wang, G. Zweig, and A. Acero, "Automated directory assistance system – from theory to practice," 2007.

[26] K. Nakamura, Q. Zhu, S. Maruoka, T. Horiyama, S. Kimura, and K. Watanabe, "Speech recognition chip for monosyllables," in *ASP-DAC*, pp. 396–399, ACM, 2001.

[27] F. Vargas, R. Fagundes, and D. Junior, "A FPGA-based Viterbi algorithm implementation for speech recognition systems," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, vol. 2, pp. 1217–1220, IEEE, 2001.

[28] J. W. Schuster, K. Gupta, and R. R. Hoare, "Speech silicon AM: an FPGA-based acoustic modeling pipeline for hidden markov model based speech recognition," in *IPDPS*, IEEE, 2006.

[29] H. Lim, K. You, and W. Sung, "Design and implementation of speech recognition on a softcore based FPGA," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol. 3, IEEE, 2006.

[30] S. Ke, Y. Hou, Z. Huang, and H. Li, "A HMM speech recognition system based on FPGA," in *Image and Signal Processing, 2008. CISP'08. Congress on*, vol. 5, pp. 305–309, IEEE, 2008.

LOW COST LIBRARY FOR PREPROCESSING OF DIGITAL SPEECH SIGNALS

[1]Daniella D. C. da Silva, [1]Elmar U. K. Melcher, [1]Joseana M. Fechine, [1,2]Benedito G. Aguiar Neto

[1]Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática
Avenida Aprígio Veloso 882, Bodocongó, 58.109-970-Campina Grande, PB, Brazil
[2]University of Washington, Tacoma, Institute of Technology
Box 358426 1900 Commerce Street, Tacoma, Washington 98402-3100
{daniella, elmar, joseana}@dsc.ufcg.edu.br, bganeto@u.washington.edu

ABSTRACT

The man-machine interaction through speech allows, in many applications, an attractive, comfortable and effective communication. However, processing requirements still make it difficult the implementation of man-machine vocal communication systems in devices with low computational power, such as mobile phone, palmtops and home equipments. This paper describes the development of a signal preprocessing library for speech recognition systems applied to devices with low computational power. Within this context, several issues have been considered, such as limited power consumption, memory and processing capacity without reducing the efficiency of the recognition process.

1. INTRODUCTION

Nowadays, electronic equipment developers have been trying to add to these products, functionalities and characteristics that attract user's interest. However, although sophisticated, such equipments still present difficulties in their use, due to the artificial way which their users interact with them. The possibility to allow the man-machine interaction through speech makes it easier and more productive, even for carriers of visual and/or motion disability since user's eyes and hands are not necessary and thus they can be available for other tasks.

This work consists of the study and optimization of Digital Processing of Speech Signals (DPSS) techniques for application in embedded systems which will result in a library for digital preprocessing of speech signals.

This paper is organized as follows. In Section 2 the concepts of Digital Processing of Speech Signals and Embedded Systems are discussed, including motivations and difficulties for man-machine vocal communication, as well as peculiarities involved in embedded systems development. In Section 3 the developed system is described. In Section 4 the results are presented and the conclusions in Section 5.

2. DIGITAL PROCESSING OF SPEECH SIGNALS AND EMBEDDED SYSTEMS

The technological progress in the area of DPSS allows the development of vocal man-machine communication systems which can be divided in three areas [1]:

1. Speech Synthesis Systems;
2. Speech Recognition Systems;
3. Speaker Recognition Systems.

Application development for vocal man-machine communication (speech or speaker recognition) consists of a pattern recognition task, in this case, speech patterns. This type of system (Fig. 1) includes two phases [3]: training and recognition. Based on the training data that consist of sentences (words or phrases), reference models are generated, to which labels are attributed, to identify each pattern (sentence or speaker). In the recognition phase, through the test data (speech signals), test patterns are obtained and compared with the models generated during the training phase. Using a decision rule, the reference pattern nearest to the test pattern is identified [1], [4].

In the preprocessing stage the goal is to reduce noise and prepare the speech signal to next stages of the recognition process. Generally this process includes the following tasks: acquisition, preemphasis, division into frames and windowing [1], [5]. It is important to

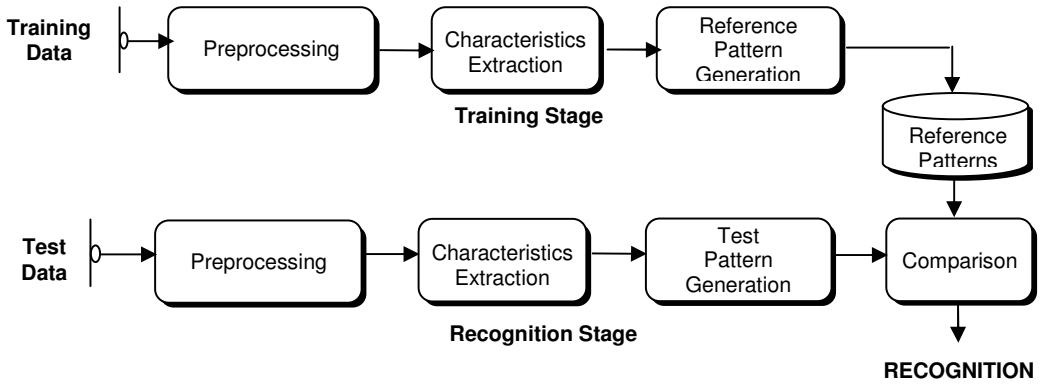


Fig. 1 – Pattern Recognition System.

highlight that the preprocessing is always present in speech or speaker recognition applications. In this regard, the possibility to have those functions already implemented in a library would turn the development process of this type of system faster and more productive.

Research in the area of Digital Processing of Speech Signals (DPSS) has been allowing the development of efficient Speech Synthesis Systems, Speech Recognition Systems and Speaker Recognition Systems. However, processing requirements still make it difficult the implementation of those systems in devices with low computational power, such as mobile phone, palmtops and home equipments. So, development of specific hardware and software for this context, also known as embedded systems, is necessary. To allow DPSS implementations in embedded systems, several issues should be considered, such as mobility, limited power consumption, low memory, storage and processing capacity, all this without lowering the efficiency of the recognition process [6], [7], [8]. New products have a shorter lifespan and delays of a few weeks in the product release can seriously reduce the expected commercial gains. The hardware project automation tends in the direction of platforms reuse and integrated components in SoC (IP Cores). Thus, the software project automation and its integration with the hardware project become the main objective to be reached for the reduction of the total project time, without sacrificing quality. Software and hardware components reuse is also essential in this context [9].

3. PROPOSED SYSTEM

Prototype developed in this work is based on the project described by Cipriano [5], which presents adaptations of the model to speech recognition, allowing their implementation in devices with low computational power. For the purpose of this implementation, some strategies are adopted. For example, fixed-point is used instead of operations with fractional numbers, mathematical functions are simplified, division and multiplication operations are substituted by bit shifting, parallelism, etc. A system diagram is presented in Fig. 2.

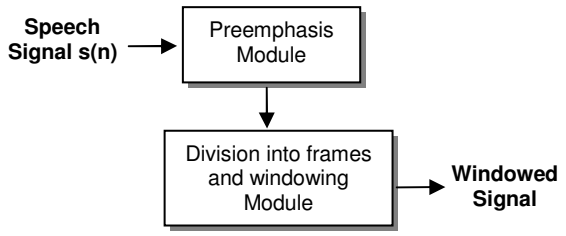


Fig. 2 – System Diagram.

The preemphasis function intends to eliminate a spectral tendency of approximately -6 dB per octave in irradiated speech from the lips. This spectral attenuation does not bring additional information and it can be eliminated through a filter application of approximately +6 dB per octave answer causing an equalization in spectrum. Preemphasis is given by [3]:

$$S_p(n) = S(n) - ap \cdot S(n-1) \quad (1)$$

Where:
$S_p(n)$ – preemphasized sample;
$S(n)$ – original sample;
ap – preemphasis factor, $0{,}9 \leq ap \leq 1$.

To simplify preemphasis filter implementation, the factor ap is replaced by the value 15/16 in Equation (1), resulting in Equation (2):

$$S_p(n) = S(n) - \frac{15}{16} S(n-1) = S(n) - S(n-1) + \frac{S(n-1)}{16} \quad (2)$$

The modification in Equation (1) implies in replacing a fractional number multiplication by a 2^n format number division, n being integer. This division can be accomplished through n bits right shift operation, in this case $n = 4$.

Generally speaking, speech signal is non-stationary. However, when the speech signal is observed over a very small interval (windows or frames), such as 16-32 msec, an interval may be found to be mostly stationary [1], [3]. Segmentation into frames consists of dividing the speech signal in segments, perfectly selected by windows or frames with perfectly defined duration [1]. Segment size is chosen considering the signal stationary limits (16-32 ms) [1]. In speech processing, the Hamming window function is typically used, which minimizes the signal discontinuities in the beginning and end of each frame, and whose equation is given in (3), where N_s is the number of samples in one frame:

$$W(n) = 0{,}54 - 0{,}46 \cdot \cos\left(\frac{2\pi \cdot n}{N_s - 1}\right), 0 \leq n \leq N_s - 1 \quad (3)$$

Windowing is made together with division into frames. In this work, these frames (Q_j) have 252 samples (23 ms) composed by three blocks (B_j), with 84 (K) samples (Fig. 3). This strategy is used to implement a superposition of 66%. So, from the second frame on, each frame includes the two last blocks from the previous frame. Fig. 3 shows the windowing process where $W(n)$ represent the Hamming window values and $S_p(n)$ is the preemphased speech signal. Each frame sample is multiplied by the corresponding window value.

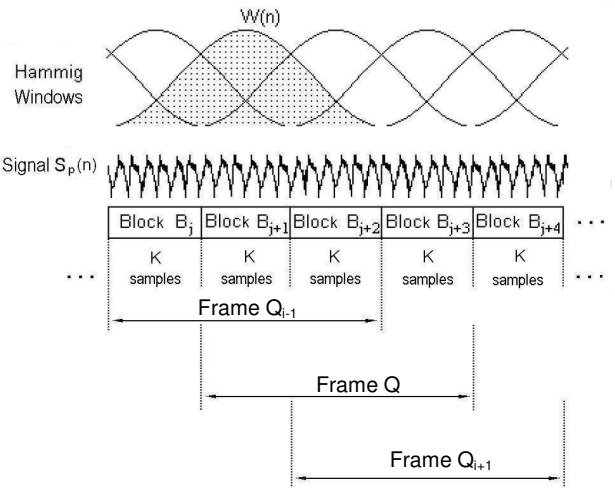


Fig. 3 – Division into frames and Windowing.

For the implementation of windowing algorithm, it was utilized a dual port R/W RAM memory which allows data reading and writing paralelly. This memory was divided into three segments with 84 samples each one. In each memory segment the blocks samples are stored, and these blocks compose the frames. Result from the multiplication between the sample of one frame and the correspondent hamming window value is stored in a register.

Samples writing in memory segments occurs in parallel to samples reading. So, while one segment is being written, another segment is being read and its samples are multiplied by hamming window values. If a same block is utilized to compose more than one frame, then it is read more than once and may not be replaced in the memory by another block while it still is necessary. The writing speed is three times slower than the reading one to avoid blocks superposition. So, while 84 samples of one memory segment are being read, 28 samples are being written in another segment.

The window values are stored into ROM memory. This allows the use of different window types, since these values could be calculated off-line. In tests, Hamming window was used with the same size of frames. Only the 126 first values are stored into memory $W(n)$ (n=0, ..., 125), and the other remainder values are obtained through the window symmetry property. The used values for the Hamming window are in interval of $0 < W(n) < 1$. As the integer part is always *0* only the fractional part is stored into ROM memory. These strategies are used to reduce memory resources.

After the implementation in a hardware description language the stages presented in Fig. 4 were needed for system prototyping.

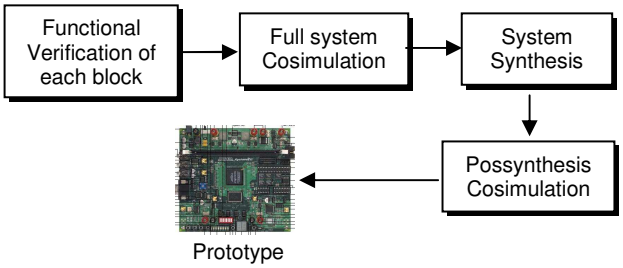


Fig. 4 – Development stages after design.

Functional verification aims to verifying all project functionalities and assure these are implemented in the specified way. This task can consume over 70% of the total development time [10], [11], and thus, tools that facilitate the fast and efficient generation of a simulation environment should be used.

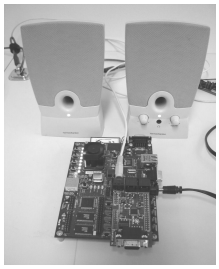
The functional verification methodology used in this work was VeriSC [11] which automatically generates a simulation environment (testbench) implemented in SystemC language that offers advantages such as transaction level, coverage-driven, self-checking and random-constraint [12].

After assuring that each block was running in the specified way, the cosimulation of the whole system was done. This stage intended to verify the correct communication among all blocks. In the synthesis stage the device to be used for the prototyping system was defined.

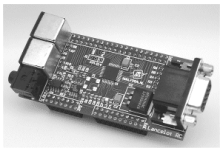
From this point on, specific device aspects were considered, for instance, logic gate delays. A new cosimulation of the whole system was necessary to assure that it continues operating correctly. After this, the final protyping stage could be accomplished in a FPGA. In this work, the Altera device Stratix II EP2S60F672C5ES was used for the prototyping system.

4. RESULTS

The database composed by ten speech signals was used as Preprocessing input, including phonemes, syllables, words and sentences. These signals were captured through a microphone and stored into Stratix II flash memory. Preemphasis output signal was connected to loudspeakers to verify the preemphasis filter effect (Fig. 5). Lancelot device was necessary to connect Stratix II device to loudspeakers because Stratix II device does not have audio output.



(a)



(b)

Fig. 5 – (a) Stratix II device connected to a loudspeaker through (b) Lancelot device.

Considering preprocessing module as unique, in Fig. 6 developed prototype diagram is presented. The system includes *Clock_Divisor, Clock_Generator, FLASH* and *PWM.*

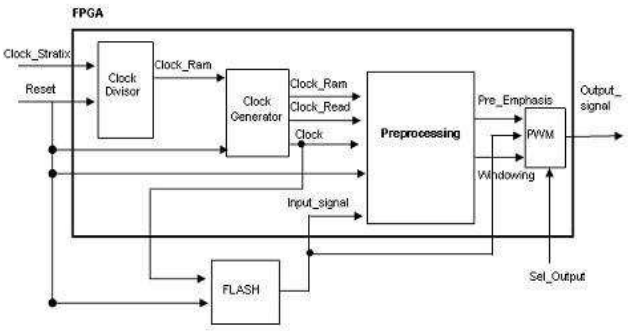


Fig. 6 – Developed Prototype Diagram.

Clock_Divisor and *Clock_Generator* are two frequency divisors. The first receives clock of Stratix II device (50 MHz) and generates the greatest frequency of system

(*Clock_Ram*). *Clock_Generator* receives *Clock_Ram* and generates the two other clocks used in the system. *FLASH* module function reads samples stored in the flash memory. And *PWM* is a D/A conversor based on Pulse Width Modulation (PWM).

Table 1 shows the preprocessing calculations. Comparison of theoretical results ($W_t(n)$ and $J_t(n)$) with simulated ones ($W(n)$ and $J(n)$) proves correct operation of the developed system. Because of fixed-point usage, in *Simulation Result* the four last digits are considered as the fractional part from the result and the others the integer part.

Table 1 – Preprocessing Calculations.

$S(n)$	$S_p(n)$	$W_t(n)$	$W(n)$	**Theorical Result** $J_t(n) = S_p(n).W_t(n)$	**Simulation Result** $J(n) = S_p(n).W(n)$
0	0	0	0	0	0
-145	-145	0.0800	0800	-11.6000	-116000
-159	-24	0.0801	0801	-1.9224	-19224
-175	-26	0.0806	0806	-2.0956	-20956
-188	-24	0.0813	0813	-1.9512	-19512
-203	-27	0.0823	0823	-2.2221	-22221
-218	-28	0.0836	0836	-2.3408	-23408
-233	-29	0.0852	0852	-2.4708	-24708

All modules were coded in VHDL resulting in 935 code lines and 407 SystemC code lines in the testbench. Functional verification of the complete preprocessing module included 5000 transactions.

After the synthesis through Altera Quartus 4.1 tool, the amount of resources necessary to library implementation was obtained. These values are presented in Table 2.

Table 2 – Library Resources.

Resource	**Utilized Amount**	**Cyclone II EP2C70 Availability**
Input pins	43	622
Output pins	51	
Bidirectional pins	24	
Logic cells	1.600	88.418
Memory bits	808.896	1.152.000
DSP Elements	2	150

Information presented in Table 2 proves that developed library requires few resources and could be implemented in Cyclone II EP2C70, one of the cheaper devices for digital signals processing from Altera.

A graph with time expended in development stages is shown in Fig. 7.

As expected, functional verification consumed 77% of the total development time, including functional verification of each block (47%), full system cosimulation (15%) and possynthesis cosimulation (15%).

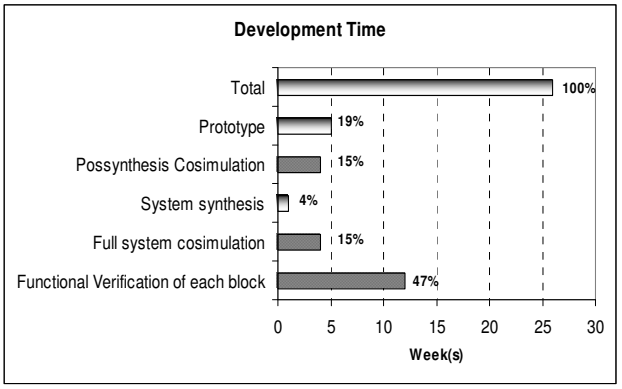


Fig. 7 – Development time.

5. CONCLUSIONS

The work result is a synthesizable and verified module. This module makes possible the construction of a Speech Preprocessing library, including preemphasis, division into frames and windowing. This library could be reused in the development of embedded speech or speaker recognition applications. The problem of different noise levels could spoil the recognition process. So, for a future work we intend to consider different noise levels, as well as the inclusion of a speech signal acquisition module.

6. REFERENCES

[1] L. R. Rabiner, R. W. Schafer "Digital processing of speech signals," *New Jersey: Prentice Hall*, 1978.

[2] W. B. Kleijn, K. K. Paliwal "Speech Coding and Synthesis," *Elsevier Science B. V.*, 1998.

[3] L. R. Rabiner, B. Juang "Fundamentals of Speech Recognition," *New Jersey: Prentice Hall*, 507p, 1993.

[4] R. Deller Jr., J. G. Proakis and J. H. L. Hansen "Discrete-time Processing of Speech Signals," *Macmillan Publishing Co*., 1993.

[5] J. L. G. Cipriano "Desenvolvimento de Arquitetura Para Sistemas de Reconhecimento Automático de Voz Baseados em Modelos Ocultos de Markov," *Universidade Federal do Rio Grande do Sul*, Porto Alegre, Brasil, 2001.

[6] R. Krishna, S. Mahlke, T. Austin "Memory system design space exploration for low-power, real-time speech recognition," *Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, p. 140 – 145, 2004.

[7] S. Nedevschi, R. K. Patra and E. A. Brewer "Hardware Speech Recognition for User Interfaces in Low Cost, Low Power Devices," *Proceedings of the 42nd annual ACM IEEE conference on Design automation*, p. 684 – 689, San Diego, California, USA. 2005.

[8] E. C. Lin, K. Yu, R. A. Rutenbar and T. Chen. "A 1000-word vocabulary, speaker-independent, continuous live-mode speech recognizer implemented in a single FPGA," *Proceedings of the 2007 ACM/SIGDA 15th international symposium on Field programmable gate arrays*, February 2007.

[9] L. Carro, F. R. Wagner "Sistemas Computacionais Embarcados", *JAI'03 – XXII Jornadas de Atualização em Informática*, Campinas, São Paulo, Brasil, 2003.

[10] J. Bergeron. "Writing Testbenches: Functional Verification of HDL Models," *Kluwer Academic Publishers*, January 2000.

[11] K. R. da Silva, E. U. K. Melcher and G. Araujo "An Automaic Testbench Generation Tool for a SystemC Functional Verification Methodology," *SBCCI2004 – 17th Symposium on Integrated Circuits and System Design*, Porto de Galinhas – PE, Brazil, 2004.

[12] Open SystemC initiative, *http://*www.systemc.org

Otimização de Técnicas de Processamento Digital de Sinais de Voz para Aplicação em Sistemas Embutidos

Daniella Dias Cavalcante da Silva, Elmar Uwe Kurt Melcher, Joseana Macêdo Fechine

Universidade Federal de Campina Grande
Departamento de Sistemas e Computação
Av. Aprígio Veloso, 882, Bodocongó
Campina Grande – PB – Brasil
{daniella, elmar, joseana}@dsc.ufcg.edu.br

Resumo

A fala é o meio de comunicação comumente utilizado pelo homem, que o distingue dos demais seres vivos, permitindo-lhe a troca de idéias, expressão de opiniões ou revelação de seu pensamento. Diante do avanço tecnológico e conseqüente surgimento de equipamentos eletrônicos cada vez mais sofisticados, a possibilidade de permitir a interação homem-máquina através da voz tornaria essa relação muito mais atrativa e produtiva, não só para a comunidade em geral, mas principalmente para portadores de deficiência visual e/ou motora. Este artigo descreve o desenvolvimento de um sistema de reconhecimento de voz otimizado para dispositivos com pequeno poder computacional, como equipamentos eletrônicos em geral, eletrodomésticos ou ainda, que possa ser utilizado para operar elevadores, portas automáticas, etc.

1. Introdução

Atualmente, fabricantes de equipamentos eletrônicos têm procurado adicionar aos seus produtos, funcionalidades e diferenciais que atraiam o interesse de seus usuários. Como exemplo, tem-se televisores conectados à Internet; máquinas fotográficas que permitem "zoom" e alteram resolução, além de se conectarem a computadores e impressoras; fornos de microondas e máquinas de lavar roupas com controle "inteligente", etc. No entanto, embora sofisticados, tais equipamentos ainda apresentam dificuldades quanto a sua utilização, devido à maneira artificial com a qual o usuário deve interagir com os mesmos.

A fala é o meio de comunicação utilizado por excelência pelo homem e que o distingue dos demais seres vivos, permitindo-lhe a troca de idéias, expressão de opiniões ou revelação de seu pensamento. Diante disso, a comunicação vocal homem-máquina torna-se bastante adequada em situações nas quais uma ou mais das seguintes condições se aplicam: as mãos do usuário estão ocupadas; mobilidade é exigida durante o processo de entrada de dados; os olhos do operador devem permanecer fixos sobre um display; um instrumento óptico ou algum objeto é rastreado; é inconveniente o uso do teclado em um ambiente, dentre outras. A possibilidade de permitir a interação homem-máquina através da voz torna essa interação mais fácil e produtiva, principalmente para portadores de deficiência motora e/ou visual, uma vez que não se faz necessária a utilização das mãos e olhos do usuário, que podem ainda, ficar disponíveis para outras tarefas.

Pesquisas na área de Processamento Digital de Sinais de Voz (PDSV) têm permitido o desenvolvimento de sistemas de Resposta Vocal, Reconhecimento de Voz e Reconhecimento de Identidade Vocal (locutor). Entretanto, requisitos de processamento ainda dificultam a implementação desses sistemas em dispositivos com pequeno poder computacional, como aparelhos celulares, *palmtops* e eletrodomésticos. Surge então, a necessidade de desenvolvimento de *hardware* e *software* específicos para esse contexto, também conhecidos como sistemas embutidos, ou sistemas embarcados. Para permitir a implementação de técnicas de PDSV em sistemas embutidos, várias questões devem ser consideradas, como mobilidade, limite de consumo de potência, baixa disponibilidade de memória, segurança e confiabilidade. Tudo isso sem comprometer a eficiência no processo de reconhecimento.

O trabalho desenvolvido consiste no estudo e otimização de técnicas de PDSV para aplicação em sistemas embutidos, que resultará no desenvolvimento de uma biblioteca de funções para pré-processamento digital de sinais de voz (pré-enfase; divisão em "frames" e janelamento). Com o intuito de validar a biblioteca, será desenvolvida uma aplicação de Reconhecimento de Palavras Isoladas Dependente do Locutor que levará em consideração as restrições quanto à capacidade de processamento e armazenamento, características de um ambiente embutido.

Este artigo está organizado conforme descrição a seguir. Na seção 2 são discutidos os conceitos sobre Processamento Digital de Sinais de Voz, em que são apresentadas algumas das motivações para a comunicação vocal homem-máquina, bem como as dificuldades encontradas nesse tipo de aplicação. Na seção 3 será apresentado o conceito de sistemas embutidos, assim como as pecularidades envolvidas no seu desenvolvimento. Na seção 4 é realizada a descrição do sistema desenvolvido e finalmente, na seção 5 é

apresentado o estado atual do trabalho e algumas considerações finais.

2. Processamento Digital de Sinais de Voz

Os primeiros trabalhos sobre máquinas que conseguiam reconhecer algumas palavras datam de 1952 [1]. Graças às descobertas de algumas propriedades da voz e das facilidades oferecidas pelos computadores digitais, nos anos 60 surgiu uma grande quantidade de trabalhos sobre o assunto [2], [3]. Posteriormente, desejou-se a criação de máquinas que, além de serem capazes de reconhecer o que estava sendo dito, também pudessem responder ao que lhes era perguntado.

Diante do crescente interesse por *software* e equipamentos que "compreendam", reconheçam e simulem a voz humana, pesquisas têm sido realizadas na área de Processamento Digital de Sinais de Voz (PDSV), buscando o desenvolvimento de técnicas que possibilitem a construção de padrões que permitam modelar, de forma eficiente, a fala, as características vocais únicas de cada locutor, bem como a produção da voz sintetizada [4], [5].

O avanço tecnológico na área de PDSV permite o desenvolvimento de sistemas de comunicação vocal homem-máquina, que podem ser divididos em três grandes áreas [3]:

1. Sistemas de Resposta vocal;
2. Sistemas de Reconhecimento de voz;
3. Sistemas de Reconhecimento de locutor.

Sistemas de resposta vocal, ou sistemas de síntese de voz, são projetados para responder a um pedido de informação utilizando mensagens faladas [6], [7].

Sistemas de resposta vocal podem ser utilizados em várias aplicações, tais como sistemas automáticos de informação de preços, de vôos, de divulgação de produtos em supermercados ou lojas especializadas, sistemas bancários, auxílio a portadores de necessidades especiais, etc. [6], [8]].

Sistemas de reconhecimento de voz, ou ainda Sistemas de Reconhecimento Automático de Voz (RAV), têm como objetivo determinar qual a palavra, a frase ou a sentença que foi pronunciada. De uma forma geral, são considerados como pertencentes a uma das seguintes categorias: Sistemas de Reconhecimento de Palavras Isoladas; Sistemas de Reconhecimento de Palavras Conectadas; Sistemas de Reconhecimento de Fala Dependente do Locutor e Sistemas de Reconhecimento de Fala Independente do Locutor [3]. A diferença entre os sistemas dependentes e os independentes de locutor é que no primeiro, o sistema é treinado para um locutor específico, enquanto que o segundo está preparado para reconhecer sentenças, independente de quem as pronuncie. Sistemas de reconhecimento de fala podem ser utilizados em diversas aplicações, dentre as quais pode-se citar: atendimento telefônico automatizado, acesso a menus de equipamentos eletrônicos (celular, fornos de microondas, aparelhos de DVD, etc.), transcrição de sentenças pronunciadas para um arquivo texto, etc [8]. Os sistemas para fala contínua (palavras conectadas) são mais difíceis de implementar do que os sistemas para fala descontínua (com pronúncia isolada das palavras), devido à dificuldade de se localizar o início e o fim de cada palavra, e também à tendência das línguas faladas de fundir o último e o primeiro fonema de duas palavras consecutivas.

Quanto aos sistemas de reconhecimento de locutor, estes podem ser de dois tipos: verificação ou identificação. O primeiro consiste em decidir se a voz é mesmo de quem alega ser, enquanto que o segundo deve ser capaz de identificar quem é o locutor, dentre um universo de locutores. Esses sistemas são muito úteis para o controle de acesso a ambientes restritos. Na área de criminalística podem ser utilizados com o mesmo propósito que hoje é dado às impressões digitais [3], [8], [9].

O desenvolvimento de aplicações na área de comunicação vocal homem-máquina (reconhecimento de fala ou locutor) consiste na realização de uma tarefa de reconhecimento de padrões, nesse caso padrões de fala. Esse tipo de sistema (Figura 1) inclui duas fases [10]: treinamento e reconhecimento. Com base nos dados de treinamento, que consistem de sentenças (palavras ou frases), são gerados os modelos de referência, aos quais são atribuídos rótulos que identificam cada padrão (sentença ou locutor). Na fase de reconhecimento, através dos dados de teste (sinais de voz) são obtidos padrões de teste que, em seguida, são comparados com os modelos gerados durante o treinamento e, utilizando-se uma regra de decisão, é identificado o que mais se assemelha ao padrão de entrada desconhecido [3], [11].

A etapa de pré-processamento é responsável por extrair do sinal de voz os dados relevantes e menosprezar a informação redundante, com a finalidade de repassar a informação de interesse para etapa seguinte. Esse processo inclui as atividades de: aquisição, pré-ênfase, separação em "frames" e janelamento [3], [12]. É importante destacar que o pré-processamento precisa ser executado independente do tipo de reconhecimento que se deseja realizar (voz ou locutor). Diante disso, a possibilidade de se ter essas funções já implementadas em uma biblioteca, tornaria o processo de desenvolvimento desse tipo de aplicação mais rápido e produtivo.

A etapa de extração de características é de extrema importância em sistemas de reconhecimento, uma vez que nesta etapa são obtidos elementos que possibilitam a geração dos padrões de treinamento e teste. A definição de um bom conjunto de características é um processo complexo e essencial para que sejam obtidos bons resultados quanto ao reconhecimento dos padrões.

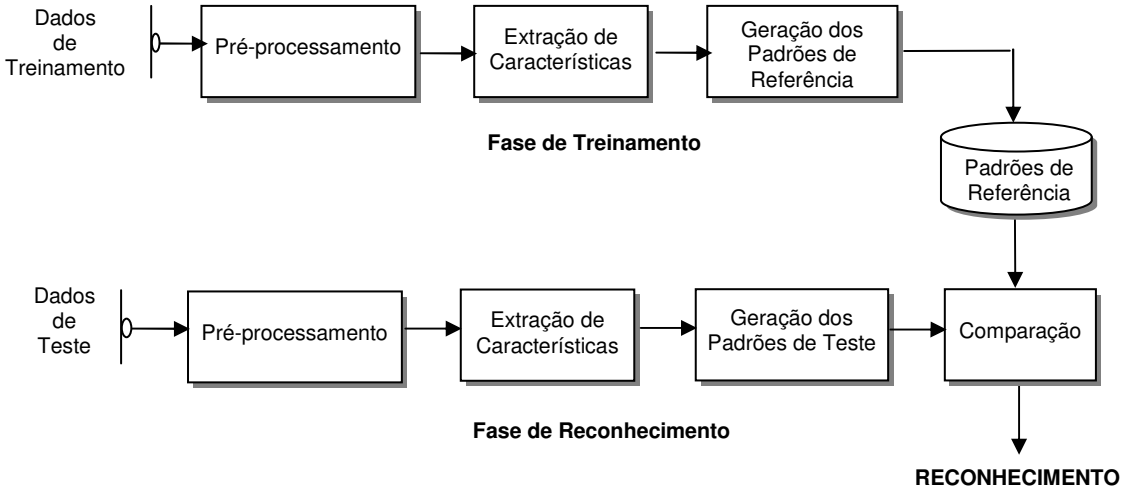


Fig. 1. Sistema de Reconhecimento de Padrões.

Durante a fase de treinamento, com base nas características extraídas, são gerados padrões de referência, os quais serão comparados com o padrão de teste na fase de reconhecimento. Na fase de teste, o padrão de teste é comparado com os padrões de referência e, utilizando-se uma regra de decisão, o locutor (ou a sentença) cujo padrão de teste é mais similar a um dado padrão de referência, é reconhecido pelo sistema.

Vale salientar que as etapas de pré-processamento e extração de características das fases de treinamento e reconhecimento devem ser equivalentes, para que seja possível uma correta comparação entre o padrão testado e o(s) padrão(ões) já conhecido(s) pelo sistema.

3. Sistemas Embutidos

É importante destacar que, quando se fala em interface homem-máquina, deve-se ficar claro que esta máquina pode ser qualquer equipamento eletrônico. No âmbito deste trabalho, pode ser um aparelho celular, um *palmtop*, uma máquina de lavar, um forno microondas, um veículo, uma máquina industrial, dentre outros equipamentos que apresentam arquitetura e limitações (capacidade de processamento e armazenamento, fonte de alimentação, etc) diferentes de um computador. Diante disso, torna-se necessário o desenvolvimento de *software* e/ou *hardware* específico para este tipo de equipamento. Surge então, o conceito de sistema embutido (embarcado), que é a combinação de *hardware* e *software*, e algumas vezes peças mecânicas, desenvolvidos para realizar uma função específica [13]. Através da implementação em *hardware* é possível alcançar uma maior eficiência e rapidez na execução de determinadas tarefas e, a partir do *software*, reduzir o tempo de desenvolvimento e aumentar a flexibilidade do sistema.

No que se refere à implementação em *hardware* de um sistema embutido, diferentes soluções podem ser adotadas. Para aplicações com produção em grande escala (mercado de consumo), o mais adequado é o uso de um SoC (*System-On-a-Chip* – sistema em uma única pastilha) [14]. A arquitetura de *hardware* de um SoC embutido pode conter um ou mais processadores, memórias, interfaces para periféricos e blocos dedicados. Tais blocos, também chamados de IP Cores (*Intellectual Property Cores*), consistem de blocos em *hardware* que executam tarefas específicas e são definidos de modo a permitir o seu reuso em diferentes sistemas [15]. Os diversos IP Cores de um determinado SoC são interligados por uma estrutura de comunicação que pode variar de um barramento a uma rede complexa (NoC – *network-on-chip*) [16].

Em aplicações com menor volume de produção, a implementação do sistema em FPGA *(Field-Programmable Gate Array*) é mais indicada. Apesar de ser viável também a implementação utilizando sistemas baseados em famílias de microprocessadores [14]. A implementação em FPGA apresenta como principal vantagem a possibilidade de modificação da estrutura de *hardware* do sistema através de um processo denominado reconfiguração, o qual permite o desenvolvimento incremental, correção de erros de projeto, além da adição de novas funções de *hardware* [15].

Ao projetar o *hardware* de um sistema embutido, linguagens de descrição de *hardware* (HDL – *Hardware Description Language*) podem ser utilizadas para realizar a descrição dos circuitos eletrônicos. Essas linguagens permitem descrever a forma como os circuitos operam, possibilitando também a simulação desses circuitos antes mesmo de sua fabricação. Ao contrário de uma linguagem de programação para *software*, a sintaxe e a semântica das HDL incluem informações para expressar ao longo do tempo (*timed*) qual será o comportamento do *hardware*. As linguagens mais populares são VHDL (VHSIC[1] *Hardware Description Language*) e Verilog, que apresentam a grande vantagem de possibilitar a sua utilização como entrada para simulação e síntese automática de circuitos descritos no nível da microarquitetura, através da utilização de ferramentas comerciais bastante difundidas. No entanto, essas linguagens não são

[1] Very High Speed Integrated Circuits

apropriadas para descrições de *software* e especificações funcionais de alto nível [14]. No intuito de sanar essa deficiência, foram realizadas tentativas com linguagens que possuíssem um maior grau de abstração, como C, C++ e Java. Nesse caso, ocorre exatamente o contrário, ou seja, a inadequação semântica dessas linguagens para descrição de aspectos de *hardware*. O uso da linguagem SystemC [17] visa solucionar essas deficiências, combinando as vantagens da popularidade de C++ e a sua adequação ao processo de geração de *software*, com uma semântica adicional (na forma de uma biblioteca de funções) apropriada para a descrição de *hardware*, através de construções como portas, sinais e relógios que sincronizam processos [14].

O projeto de sistemas embutidos é bastante complexo, por envolver conceitos pouco analisados pela computação de propósito geral [4], [5]. Por exemplo, as questões de mobilidade, limite de consumo de potência, a baixa disponibilidade de memória, a necessidade de segurança e confiabilidade, a possibilidade de funcionamento em uma rede de comunicação, e o curto tempo de projeto tornam o desenvolvimento de sistemas computacionais embutidos uma área de pesquisa bastante abrangente [14]. Além disso, novos produtos têm uma vida cada vez mais curta, e atrasos de poucas semanas no lançamento de um produto podem comprometer seriamente os ganhos esperados. Com a automação do projeto de hardware caminhando na direção do reuso de plataformas e de componentes integráveis em SoC (IP *Cores*), a automação do projeto de *software* e sua integração com o projeto de *hardware* se torna o principal objetivo a ser alcançado para a diminuição do tempo total de projeto, sem sacrifício na qualidade da solução [14]. Também é essencial o reuso de componentes de *software*, de modo que o desenvolvimento do sistema concentre-se apenas na configuração e integração desses componentes [14].

4. Sistema Proposto

Algoritmos com bom desempenho, em termos de taxa de reconhecimento, demandam um certo poder computacional, o que já se tem disponível em máquinas tipo PC (*Personal Computer*) em tempo real. No entanto, em sistemas embutidos dependentes de baterias, como um telefone celular, o desafio não representa apenas o desenvolvimento de *software*, mas também de *hardware* que se adapte às características inerentes de tais dispositivos (baixo consumo de energia, baixo poder de processamento, etc.), [4][5]. Algumas técnicas de programação permitem a geração de um código mais eficiente no que se refere à quantidade de operações a serem executadas pelo processador (p.ex.: substituição de uma multiplicação por deslocamento de bits), o que conseqüentemente proporciona menor consumo de potência e energia [18]. Sendo assim, através da otimização de algoritmos, utilização de estruturas de *hardware* específicas e um *pipeline* de instruções eficiente, é possível aumentar a velocidade, reduzir os recursos necessários, sem diminuir a taxa de reconhecimento. Dentro desse contexto, muitos trabalhos vêm sendo desenvolvidos para o reconhecimento de palavras isoladas em sistemas embutidos [4], [5], [12][20], apresentando redução do número de acessos à memória, paralelismo de operações, economia de recursos lógicos necessários para implementação, etc. No entanto, a maiorias dos trabalhos não realizam a etapa de verificação funcional, que será descrita mais adiante, ou ainda não realizam testes efetivos em algum dispositivo de *hardware*

O protótipo desenvolvido neste trabalho têm como base o projeto descrito por Cipriano em [12], que apresenta adequações dos modelos necessários ao processo de reconhecimento de voz para implementação em *hardware*. Para isso, é feita a utilização de ponto-fixo nas operações com número fracionários, simplificações nas funções matemáticas, substituição de operações de divisão e multiplicação por deslocamentos, paralelismo, dentre outras. O diagrama do sistema em desenvolvimento é apresentado na Figura 2.

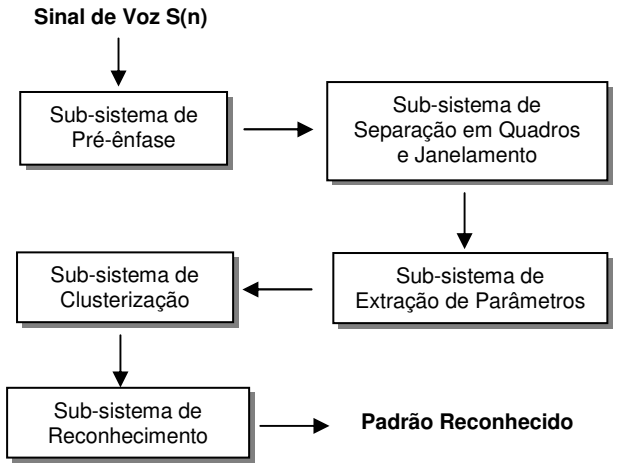


Fig. 2. Diagrama do Sistema Proposto.

A função da pré-ênfase é eliminar uma tendência espectral de aproximadamente -6dB/oitava na fala irradiada dos lábios. Essa distorção espectral não traz informação adicional e pode ser eliminada através da aplicação de um filtro, de resposta aproximadamente +6dB/oitava, que ocasiona um nivelamento no espectro. A descrição matemática da pré-ênfase é dada por [24]:

$$S_p(n) = S(n) - ap.S(n-1) \qquad \textbf{(1)}$$

Sendo:
$S_p(n)$ – amostra pré-enfatizada;
$S(n)$ – amostra original;
ap – fator de pré-ênfase, $0{,}9 \leq ap \leq 1$.

No intuito de simplificar a implementação do filtro de pré-ênfase, na Equação (1), o fator ap é substituído pelo valor 15/16:

$$Sp(n) = S(n) - \frac{15}{16}S(n-1) = S(n) - S(n-1) + \frac{S(n-1)}{16} \qquad \textbf{(2)}$$

A modificação efetuada na Equação (1) implica na substituição de uma multiplicação por um número fracionário por uma divisão por um número do tipo 2^n, sendo n inteiro, e que pode ser realizada através de um simples deslocamento de n bits, nesse caso $n = 4$.
Um sinal de voz possui características estatísticas que variam fortemente com o tempo, só podendo ser considerado estacionário em intervalos muito pequenos. A separação em quadros consiste em particionar o sinal de voz em segmentos, selecionados por janelas ou "frames" de duração perfeitamente definida [4]. O tamanho desses segmentos é escolhido dentro dos limites de estacionariedade do sinal (duração média de 10 a 20 ms) [3]. No sistema proposto o algoritmo de janelamento utlizado foi o de *Hamming*, que minimiza as descontinuidades no início e final de cada quadro, e cuja equação é dada por:

$$0{,}54 - 0{,}46 \cdot \cos\left(\frac{2\pi \cdot n}{Ns - 1}\right), 0 \le n \le Ns - 1 \quad \textbf{(3)}$$

O janelamento é realizado juntamente com a separação dos quadros, que são formados por três blocos, de 84 amostras cada, totalizando 252 amostras para cada quadro. Cada amostra do quadro é multiplicada pelo valor correspondente da janela. A utilização de 252 amostras para cada quadro visa otimizar a implementação em *hardware* dessa etapa, uma vez que o número 252 facilita a utilização de ponto-fixo e a superposição entre quadros (Figura 3). O uso de ponto-fixo permite a realização das operações de multiplicação e divisão de números fracionários através de simples deslocamentos para esquerda e para direita, respectivamente.

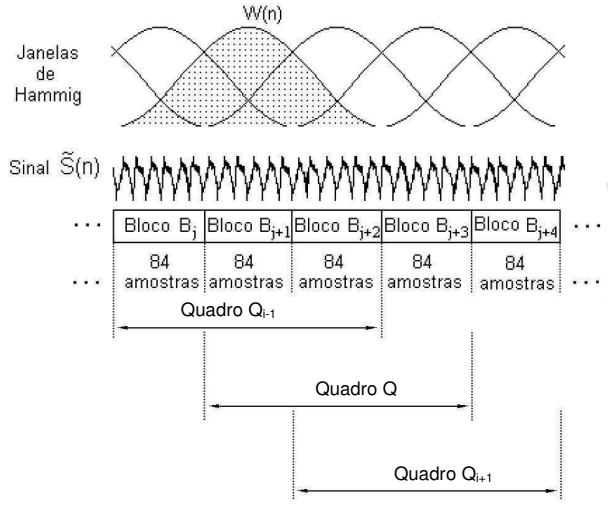


Fig. 3. Divisão em Quadros e Janelamento.

Depois da aplicação da janela de H*amming*, é realizada a extração de parâmetros, por exemplo, os parâmetros mel-cepstrais de cada quadro [14].
O subsistema de reconhecimento, que realizará a identificação da palavra falada, tem como função o casamento dos padrões de teste com os padrões armazenados. Neste trabalho será utilizada a quantização vetorial para a construção de cada padrão [14].
Após a implementação/simulação do sistema, será realizada a etapa de prototipação. As etapas previstas para obtenção do protótipo são apresentadas na Figura 4.

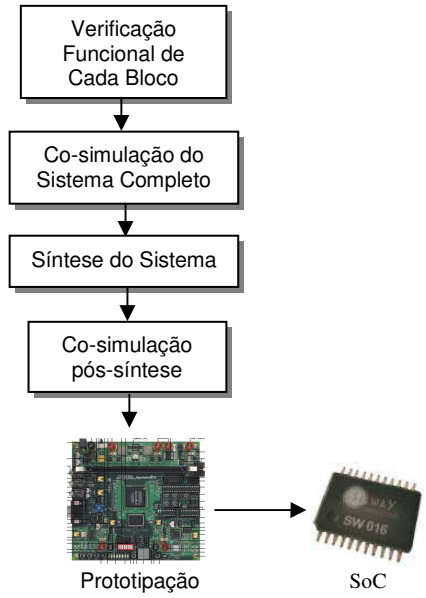


Fig. 4. Etapas de desenvolvimento.

A etapa de verificação funcional tem o objetivo de verificar todas as funcionalidades de projeto e assegurar que estas estão ocorrendo da maneira especificada. Esta tarefa chega a consumir cerca de 70% do tempo total de desenvolvimento [22], [23]. Diante disso, ferramentas que possibilitem a geração rápida e eficiente de um ambiente de simulação podem e devem ser utilizadas. Para tal, foi utilizada a ferramenta VeriSC [24] que realiza a geração automática de um ambiente de simulação (*testbench*) implementado na linguagem SystemC, que oferece as vantagens apresentadas na Seção 3, como por exemplo, a inclusão de nível de transação, cobertura dirigida, checagem automática e construção de verificação funcional randômica.
Após assegurar que cada bloco está executando da maneira especificada, será realizada a etapa de co-simulação e pós-síntese de todo o sistema, de modo a verificar a correta comunicação entre todos os blocos. Na etapa de síntese é definido qual o dispositivo que será utilizado para prototipação do sistema, o que implica que, a partir de então, aspectos específicos do dispositivo, como por exemplo, atrasos das portas lógicas, passam a ser considerados durante a execução, tornando necessária uma nova co-simulação de todo o sistema, de modo a assegurar a continuação de seu correto funcionamento. Vencida essa etapa, pode-se passar para etapa final de prototipação. Neste trabalho foi utilizada a placa Cyclone II EP2C35 DSP da Altera [25] para prototipação do sistema.

5. Conclusões

Atualmente o trabalho encontra-se no final da etapa de verificação funcional, que como foi dito anteriormente, corresponde a 70% do processo de desenvolvimento. O resultado deste trabalho será a obtenção de um módulo sintetizável que possibilitará a construção de um IP *Core* de pré-processamento de voz, incluindo as funções de pré-ênfase, divisão em quadros e janelamento. Tal módulo poderá ser reutilizado no desenvolvimento de futuras aplicações embutidas de reconhecimento de voz, ou mesmo de locutor, proporcionando uma economia considerável no tempo total do projeto. O alvo dessas aplicações poderá ser desde eletrodomésticos e equipamentos eletrônicos a elevadores e portas, operados por comandos de voz, facilitando a utilização dos mesmos por portadores de necessidades especiais, principalmente visuais e motoras.

Como trabalhos futuros pretende-se tratar a situação em que haja diferentes níveis de ruído ambiente entre a etapa de treinamento e reconhecimento, além da adição de um módulo de aquisição do sinal de voz. O problema na diferença de ruído ambiente pode comprometer a eficiência do reconhecimento. Já existem algoritmos que visam resolver essa situação, no entanto projetados para arquitetura PC. Quanto ao módulo de aquisição, o mesmo facilitará a implantação do sistema em dispositivos que não ofereçam a funcionalidade de conversão analógico-digital da voz.

Referências

[1] DAVIS et al, K. H. *Automatic Recognition of Spoken Digitis*. Journal of the Acoustical Society of America, v. 24, n. 6, p. 637-642, 1952.

[2] KOENIG, W. *The Sound Spectrograph*. Journal of the Acoustical Society of America, v. 17, p. 19-49, 1946.

[3] RABINER, L. R.; SCHAFER, R. W. *Digital processing of speech signals*. New Jersey: Prentice Hall, 1978.

[4] KRISHNA, R., MAHLKE, S., AUSTIN, T., *Memory system design space exploration for low-power, real-time speech recognition*. Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, p. 140 – 145, 2004.

[5] NEDEVSCHI, S., PATRA, R. K., BREWER, E. A. *Hardware Speech Recognition for User Interfaces in Low Cost, Low Power Devices*. Proceedings of the 42nd annual conference on Design automation, p. 684 – 689, San Diego, California, USA. 2005.

[6] KLEIJN, W. B. and PALIWAL, K. K., *Speech Coding and Synthesis*. Elsevier Science B. V., 1998.

[7] LEMMETTY, S. *Review of Speech Synthesis Technology*. Disponível em www.acoustics.hut.fi/~slemmett/dippa/chap1.html. Acessado em dezembro de 2005.

[8] GUILHOTO, P. J. dos S., ROSA, S. P. C. de S., *Reconhecimento de Voz*. Faculdade de Ciências e Tecnologia da Universidade de Coimbra, 2001.

[9] CAMPBELL, J. P., *Speaker Recognition: A Tutorial*. Proceeedings of the IEEE, v. 85, 1997.

[10] FECHINE, J. M *Reconhecimento automático de identidade vocal utilizando modelagem híbrida: Paramétrica e Estatística*. 2000. 212 f. Tese (Doutorado em Engenharia Elétrica) – Universidade Federal de Campina Grande, Campina Grande, 2000.

[11] DELLER Jr. R., PROAKIS, J. G., and HANSEN, J. H. L., *Discrete-time Processing of Speech Signals*. Macmillan Publishing Co., 1993.

[12] CIPRIANO, J. L. G., *Desenvolvimento de Arquitetura Para Sistemas de Reconhecimento Automático de Voz Baseados em Modelos Ocultos de Markov*. 123 f. Tese (Doutorado em Ciência da Computação) – Universidade Federal do Rio Grande do Sul, Porto Alegre, 2001.

[13] BARR, M. *Programming Embedded Systems in C and C++*. Sebastopol. CA, O'Reilly & Associates.

[14] CARRO, L.; WAGNER, F. R. *Sistemas Computacionais Embarcados*. In: JAI'03 – XXII Jornadas de Atualização em Informática, 2003, Campinas.

[15] MORAES, F., CALAZANS, N. MOLLER, L. BRIAO, E., CARVALHO, E. *Dynamic and Partial Reconfiguration in FPGA SoCs: Requirements Tools and a Case Study*. In: ROSENSTIEL, Wolfgang. Reconfigurable Computing. New York, USA. 2004.

[16] DE MICHELI, G., BENINI, L. *Networks-on-Chip: A New Paradigm for Systems-on-Chip Design*. DATE'02 – Design, Automation and Test in Europe, Paris, França, Mar. 2002. Proceedings, IEEE Computer Society Press, 2002.

[17] Open SystemC initiative, *http://www.systemc.org*

[18] HON, Hsiao-Wuen. *A Survey of Hardware Architectures Designed for Speech Recognition*. Technical Report CMU-CS-91-169, 1991.

[19] VASSALI, M. R.; DE SEIXAS, J. M.; ESPAIN, C. *Reconhecimento de Voz em Tempo Real Baseado na Tecnologia dos Processadores Digitais de Sinais*. In: XVIII Simpósio Brasileiro De Telecomunicações, 2000.

[20] ROCHA, O. P., THOMÉ, A. C. G., BARROS, S. R. R., *Reconhecimento de voz utilizando Wavelet e Classificador Neural*. In: VII Congresso Brasileiro de Redes Neurais, Natal, RN, 2005.

[21] RABINER, L.; JUANG, B. - H. *Fundamentals of Speech Recognition*. New Jersey: Prentice Hall, 1993. 507p.

[22] BERGERON, J., *Functional Verification of HDL models.* Kluwer Academic Publishers, Second Edition, 2002.

[23] J. Bergeron. *Writing Testbenches: Functional Verification of HDL Models*. Kluwer Academic Publishers, January 2000.

[24] DA SILVA, K. R. G.; MELCHER, E. U. K.; ARAUJO, G. *An Automaic Testbench Generation Tool for a SystemC Functional Verification Methodology*. In: SBCCI2004 – 17th Symposium on Integrated Circuits and System Design, Porto de Galinhas – PE, 2004.

[25] Altera Corporation. http://www.altera.com

Referências Bibliográficas

[1] L. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*. Englewood Cliffs, NJ: Prentice-Hall, 1978. xiii, 1, 2, 10, 11, 12, 14, 15, 16, 18, 67, 93

[2] W. Chou and B. Juang, *Pattern recognition in speech and language processing*. CRC, 2003. xiii, 1, 2, 3, 8, 9, 10, 24, 25, 33, 67

[3] T. Lee, "Speech analysis," 2005. xiii, 13

[4] T. Hirsimaki, "A review: Decision trees in speech recognition," *Helsinki University of Technology*, 2003. xiii, 26

[5] J. Fechine, "Reconhecimento Automático de Identidade Vocal Utilizando Modelagem Híbrida: Paramétrica e Estatística," 2000. xiii, 1, 18, 19, 45, 46, 47, 49, 89

[6] W. Walker, W. Walker, P. Lamere, P. Lamere, P. Kwok, P. Kwok, B. Raj, B. Raj, R. Singh, R. Singh, R. Gouvea, R. Gouvea, P. Wolf, P. Wolf, J. Woelfel, J. Woelfel, W. Walker, P. Lamere, P. Kwok, and S. Microsystems, "Sphinx-4: A flexible open source framework for speech recognition," tech. rep., 2004. xiv, 71

[7] W. Minker and S. Bennacef, "Speech and human'machine dialog," *Computational Linguistics*, vol. 31, no. 1, pp. 157–158, 2005. 1

[8] F. Casacuberta, M. Federico, H. Ney, and E. Vidal, "Recent efforts in spoken language translation," vol. 25, no. 3, pp. 80–88, 2008. 1, 8, 33

[9] P. Fung and T. Schultz, "Multilingual spoken language processing," vol. 25, no. 3, pp. 89–97, 2008. 1, 2, 8, 25, 33, 68

[10] R. Krishna, S. A. Mahlke, and T. M. Austin, "Memory system design space exploration for low-power, real-time speech recognition," in *CODES+ISSS* (A. Orailoglu, P. H. Chou, P. Eles, and A. Jantsch, eds.), pp. 140–145, ACM, 2004. 1, 3, 4, 50, 128

[11] S. Nedevschi, R. K. Patra, and E. A. Brewer, "Hardware speech recognition for user interfaces in low cost, low power devices," in *DAC* (W. H. J. Jr., G. Martin, and A. B. Kahng, eds.), pp. 684–689, ACM, 2005. 1, 3, 4, 50, 128

[12] M. F. BenZeghiba and H. Bourlard, "On the combination of speech and speaker recognition," 2003. 1

[13] J. R. Deller, Jr., J. G. Proakis, and J. H. Hansen, *Discrete Time Processing of Speech Signals*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1993. 1, 12, 34

[14] S. Furui, "Cepstral analysis technique for automatic speaker verification," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, pp. 254–272, Apr. 1981. 1, 10, 15

[15] J. CIPRIANO, *Desenvolvimento de Arquitetura Para Sistemas de Reconhecimento Automático de Voz Baseados em Modelos Ocultos de Markov. 123 f.* PhD thesis, Tese (Doutorado em Ciência da Computação)–Universidade Federal do Rio Grande do Sul, Porto Alegre, 2001. 1, 4, 89, 93, 128

[16] S. Furui, "50 years of progress in speech and speaker recognition," in *Proceedings of the 10th International Conference on Speech and Computer (SPECOM)*, pp. 1–9, 2005. 2, 8

[17] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*. Prentice-Hall, 1993. 2, 38, 95

[18] A. Waibel and C. Fugen, "Spoken language translation," vol. 25, no. 3, pp. 70–79, 2008. 2, 3, 8

[19] M. Gilbert and J. Feng, "Speech and language processing over the web," *Signal Processing Magazine, IEEE*, vol. 25, no. 3, pp. 18–28, 2008. 2, 8

[20] M. Ostendorf, B. Favre, R. Grishman, D. Hakkani-Tur, M. Harper, D. Hillard, J. Hirschberg, H. Ji, J. G. Kahn, Y. Liu, S. Maskey, E. Matusov, H. Ney, A. Rosenberg, E. Shriberg, W. Wang, and C. Woofers, "Speech segmentation and spoken document processing," vol. 25, no. 3, pp. 59–69, 2008. 2, 3, 33

[21] H. Hon, "A survey of hardware architectures designed for speech recognition," tech. rep., Technical Report CMU-CS-91-169, Carnegie Mellon University, School of Computer Science, 1991. 3

[22] R. Zurawski, *Embedded systems handbook*. CRC Press, 2006. 3

[23] W. Wolf, *Computers as components: principles of embedded computing system design*. Morgan Kaufmann, 2001. 4

[24] F. Wagner and L. Carro, "Sistemas Computacionais Embarcados," *Livro das Jornadas de Atualização em Informática*, 2003. 4, 50, 51, 52, 53, 54

[25] J. Shandle and G. Martin, "Making embedded software reusable for SoCs," *EEDesign, March*, vol. 1, 2002. 4

[26] C. Kim and S. Lee, "A digital chip for robust speech recognition in noisy environment," vol. 2, pp. 1089–1092, 2002. 4, 57, 128

[27] H. Lim, K. You, and W. Sung, "Design and implementation of speech recognition on a softcore based FPGA," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol. 3, IEEE, 2006. 4, 58, 86, 128

[28] E. C. Lin, K. Yu, R. A. Rutenbar, and T. Chen, "A 1000-word vocabulary, speaker-independent, continuous live-mode speech recognizer implemented in a single FPGA," in *FPGA* (A. DeHon and M. Hutton, eds.), pp. 60–68, ACM, 2007. 4, 58, 70, 86, 128

[29] K. Miura, H. Noguchi, H. Kawaguchi, and M. Yoshimoto, "A low memory bandwidth Gaussian mixture model (GMM) processor for 20,000-word real-time speech recognition FPGA system," in *ICECE Technology, 2008. FPT 2008. International Conference on*, pp. 341–344, IEEE, 2008. 4, 59, 86, 128

[30] E. C. Lin and R. A. Rutenbar, "A multi-fpga 10x-real-time high-speed search engine for a 5000-word vocabulary speech recognizer," in *FPGA* (P. Chow and P. Y. K. Cheung, eds.), pp. 83–92, ACM, 2009. 4, 67, 68, 70, 128

[31] H. Zhou and X. Han, "Design and implementation of speech recognition system based on field programmable gate array," 2009. 4, 67, 89, 128

[32] Y. kyu Choi, K. You, J. Choi, and W. Sung, "A real-time fpga-based 20000-word speech recognizer with optimized dram access," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 57, pp. 2119 –2131, aug. 2010. 4, 67, 68, 89, 128

[33] R. Veitch, L.-M. Aubert, R. Woods, and S. Fischaber, "Acceleration of hmm-based speech recognition system by parallel fpga gaussian calculation," in *Programmable Logic Conference (SPL), 2010 VI Southern*, pp. 197 –200, march 2010. 4, 67, 68, 86, 128

[34] R. W. Richard Veitch, Louis-Marie Aubert and S. Fischaber, "Fpga implementation of a pipelined gaussian calculation for hmm-based large vocabulary speech recognition," *International Journal of Reconfigurable Computing*, vol. 2011, 2011. 4, 67, 68, 79, 86, 128

[35] R. Veitch, R. Woods, and L.-M. Aubert, "Gpu acceleration of automated speech recognition for mobile devices," in *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on*, pp. 823 –828, july 2011. 4, 128

[36] O. Cheng, W. Abdulla, and Z. Salcic, "Hardware-software codesign of automatic speech recognition system for embedded real-time applications," *Industrial Electronics, IEEE Transactions on*, vol. 58, pp. 850 –859, march 2011. 4, 67, 68, 128

[37] D. SILVA, "Desenvolvimento de um IP Core de pré-processamento digital de sinais de voz para aplicação em sistemas embutidos," Master's thesis, Dissertação (Mestrado em Informática) UFCG, Campina Grande, 2006. 5, 93

[38] R. T. Tevah, *Implementação de um sistema de reconhecimento de fala contínua com amplo vocabulário para o português brasileiro*. PhD thesis, 2006. 8, 25, 33, 68, 72, 129

[39] Y.-Y. Wang, D. Yu, Y.-C. Ju, and A. Acero, "An introduction to voice search," vol. 25, no. 3, pp. 28–38, 2008. 8, 25, 30, 33, 68

[40] D. O'shaughnessy, *Speech communications: human and machine*. New York: IEEE Press, 2000. 10, 12, 20, 95

[41] D. J. Kershaw, A. J. Robinson, and S. J. Renals, "The 1995 abbot hybrid connectionist-hmm large-vocabulary recognition system," June 03 1996. 10, 67

[42] P. Woodland, M. Gales, D. Pye, and V. Valtchev, "The HTK large vocabulary recognition system for the 1995 ARPA H3 task," 1996. 10, 67

[43] N. Zheng and P. Ching, "Using haar transformed vocal source information for automatic speaker recognition," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, vol. 1, IEEE, 2004. 10, 67

[44] B. Atal, "The history of linear prediction," *IEEE Signal Processing Magazine*, vol. 23, no. 2, pp. 154–161, 2006. 10, 67

[45] G. Garau and S. Renals, "Combining spectral representations for large-vocabulary continuous speech recognition," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 16, no. 3, pp. 508–518, 2008. 10, 19, 67

[46] R. Dias, "Normalização de locutor em sistema de reconhecimento de fala," Master's thesis, Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, Brasil, 2000. 10, 14, 22, 34, 39, 40, 67

[47] C. Lee, D. Hyun, E. Choi, J. Go, and C. Lee, "Optimizing feature extraction for speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 1, pp. 80–87, 2003. 10, 67

[48] J. W. Picone and S. Member, "Signal modeling techniques in speech recognition," 1993. 11, 12, 14, 20, 93

[49] R. V. Andreão, *Implementação em tempo real de um sistema de reconhecimento de dígitos conectados*. PhD thesis, 2001. 14, 72, 129

[50] W. C. A. Costa, *Reconhecimento de Fala Utilizando Modelos de Markov Escondidos (HMM's) de Densidades Contínuas*. PhD thesis, 1994. 14, 15, 45, 47, 48, 89

[51] R. Chengalvarayan, "Hierarchical subband linear predictive cepstral (HSLPC) features for HMM-based speech recognition," in *Acoustics, Speech, and Signal Processing, 1999. ICASSP'99. Proceedings., 1999 IEEE International Conference on*, vol. 1, pp. 409–412, IEEE, 1999. 15

[52] R. Mammone, X. Zhang, and R. Ramachandran, "Robust speaker recognition: A feature-based approach," *Signal Processing Magazine, IEEE*, vol. 13, no. 5, p. 58, 1996. 15, 19

[53] B. S. Atal and S. L. Hanauer, "Speech analysis and synthesis by linear prediction of the speech wave," *J. Acoust. Soc. Amer.*, vol. 50, no. 2, pp. 637–655, 1971. 15

[54] J. Makhoul, "Linear prediction: A tutorial review," *Proceedings of the IEEE*, vol. 63, pp. 561–580, 1975. 15

[55] M. Vieira, "Módulo frontal para um sistema de reconhecimento automático de voz," 1989. 16, 17, 18

[56] H. Tolba and D. O'Shaughnessy, "Voiced-Unvoiced Classification Using The First Mel Frequency Cepstral Coefficient," *International Conference on Speech Processing, ICSP'97*, vol. 1, pp. 137–142, 1997. 18

[57] S. S. and V. J., "The relation of pitch of frequency: a revised scale [j]," *Am J Psychol*, vol. 53, pp. 329–353, 1940. 20

[58] A. Møller, *Auditory Physiology*. Academic Press, New York, 1983. 21

[59] S. Furui, "Speaker-independent isolated word recognition based on emphasized spectral dynamics," in *Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1986. 24, 115

[60] S. F. Bing-Hwang Juang, "Automatic recognition and understanding of spoken language - a first step toward natural human-machine communication," vol. 88, pp. 1142–1165, 2000. 24

[61] L. Deng and D. O'Shaughnessy, *Speech processing: a dynamic and optimization-oriented approach*. CRC, 2003. 24, 27, 28, 29, 30

[62] M. RAJMAN, *Speech and Language Engineering*. EPFL Press, 2007. 25, 68

[63] D. Yu, Y.-C. Ju, Y.-Y. Wang, G. Zweig, and A. Acero, "Automated directory assistance system – from theory to practice," 2007. 25, 68

[64] R. Hu and Y. Zhao, "Knowledge-based adaptive decision tree state tying for conversational speech recognition," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 15, no. 7, pp. 2160–2168, 2007. 26, 27, 31

[65] L. R. Bahl, P. F. Brown, P. V. de Souza, and R. L. Mercer, "A tree-based statistical language model for natural language speech recognition," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. ASSP-37, no. 7, p. 1001, 1989. 27

[66] J. Xue and Y. Zhao, "Random forests of phonetic decision trees for acoustic modeling in conversational speech recognition," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 16, no. 3, pp. 519–528, 2008. 27

[67] R. Rosenfeld, "Two decades of statistical language modeling: Where do we go from here?," June 14 2000. 28, 29, 30

[68] J. K. Baker, "Trainable grammars for speech recognition," in *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pp. 547–550, 1979. 28

[69] F. Jelinek, J. Lafferty, and R. Mercer, "Basic methods of probabilistic context free grammars," *Speech Recognition and Understanding: Recent Advances, Trends, andApplications*, vol. 75, pp. 345–360, 1992. 28

[70] S. F. Chen, K. Seymore, and R. Rosenfeld, "Topic adaptation for language modeling using unnormalized exponential models," Aug. 12 1998. 29

[71] S. F. Chen and R. Rosenfeld, "A survey of smoothing techniques for me models," vol. 8, no. 1, pp. 37–50, 2000. 29

[72] D. Beeferman, A. L. Berger, and J. D. Lafferty, "A model of lexical attraction and repulsion," in *ACL*, pp. 373–380, 1997. 29

[73] Citeseer, *Compact maximum entropy language models*, 1999. 29

[74] S. Khudanpur and J. Wu, "A maximum entropy language model integrating n-grams and topic dependencies for conversational speech recognition," in *Proc. Conf. IEEE Int Acoustics, Speech, and Signal Processing ICASSP'99*, vol. 1, pp. 553–556, 1999. 29

[75] J. Wu and S. Khudanpur, "Combining nonlocal, syntactic and N-gram dependencies in language modeling," Apr. 28 1999. 29

[76] R. Kuhn, "Speech recognition and the frequency of recently used words: a modified markov model for natural language," *COLING-88*, pp. 348–350, 1988. 30

[77] J. Kupiec, "Probabilistic models of short and long distance word dependencies in running text," in *Proceedings of the workshop on Speech and Natural Language*, pp. 290–295, Association for Computational Linguistics, 1989. 30

[78] R. Kuhn and R. D. Mori, "A cache-based natural language model for speech recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-12, pp. 570–583, June 1990. 30

[79] R. Kuhn and R. de Mori, "Corrections to: A cache-based language model for speech recognition," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 14, no. 6, pp. 691–692, 1992. 30

[80] F. Jelinek, B. Mérialdo, S. Roukos, and M. Strauss, "A dynamic language model for speech recognition," in *HLT*, Morgan Kaufmann, 1991. 30

[81] R. Kneser and V. Steinbiss, "On the dynamic adaptation of stochastic language models," in *icassp*, pp. 586–589, IEEE, 1993. 30

[82] Y. Gao, B. Ramabhadran, J. Chen, H. Erdogan, H. Erdo, and M. Picheny, "Innovative approaches for large vocabulary name recognition," 2001. 30, 31

[83] B. Ramabhadran, L. Bahl, P. DeSouza, and M. Padmanabhan, "Acoustics-only based automatic phonetic baseform generation," in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, vol. 1, pp. 309–312, IEEE, 1998. 30

[84] F. Bechet, R. de Mori, and G. Subsol, "Dynamic generation of proper name pronunciations for directory assistance," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP'02)*, vol. 1, 2002. 31

[85] J. Martins, *Avaliação de diferentes técnicas para reconhecimento de fala*. PhD thesis, 1997. 33, 37, 38, 39, 40

[86] L. R. Rabiner and B. H. Juang, "An introduction to hidden markov models," *ASSP Magazine*, pp. 4–16, Jan. 1986. 33, 44

[87] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989. 33, 41, 42, 44, 45, 46, 47, 48, 49, 89

[88] S. Young, "Large vocabulary continuous speech recognition: A review," *IEEE Signal Processing Magazine*, vol. 13, no. 5, pp. 45–57, 1996. 33

[89] W. H. Abdulla, "HMM-based techniques for speech segments extraction," *Scientific Programming*, vol. 10, no. 3, pp. 221–239, 2002. 33

[90] W. H. Abdulla and N. Kasabov, "Reduced feature-set based parallel CHMM speech recognition systems," *Inf. Sci*, vol. 156, no. 1-2, pp. 21–38, 2003. 33

[91] C. A. Ynoguti, *Reconhecimento de fala contínua usando modelos ocultos de markov*. PhD thesis, 1999. 36, 71, 72, 128

[92] K. LEE, H. HON, and R. REDDY, "An overview of the sphinx speech recognition system," 1990. 38, 70

[93] B. H. Juang and L. R. Rabiner, "Hidden Markov Models for speech recognition," *Technometrics*, vol. 33, no. 3, pp. 251–272, 1991. 38

[94] L. R. Rabiner and S. E. Levinson, "A speaker-independent, syntax-directed, connectedd word recognition system based on hidden Markov models and level building," *ieeeassp*, vol. 33, no. 3, pp. 561–573, Jun., 1985. 46

[95] M. Savic and S. Gupta, "Variable parameter speaker verification system based on hidden Markov modeling," in *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, pp. 281–284, IEEE, 1990. 46

[96] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimal decoding algorithm," *IEEE Trans. Information Theory*, vol. IT-13, pp. 260–269, Apr. 1967. 47

[97] S. Maurer, "A survey of embedded systems programming languages," *Potentials, IEEE*, vol. 21, pp. 30 –34, apr/may 2002. 50

[98] F. Moraes, N. Calazans, L. M
"oller, E. Brião, and E. Carvalho, "Dynamic and partial reconfiguration in FPGA SoCs: requirements tools and a case study," *New Algorithms, Architectures and Applications for Reconfigurable Computing*, pp. 157–168, 2005. 50

[99] G. D. Micheli and L. Benini, "Networks on chip: A new paradigm for systems on chip design," in *DATE*, pp. 418–419, IEEE Computer Society, 2002. 50

[100] J. Bergeron, *Writing testbenches: functional verification of HDL models*. Springer Netherlands, 2003. 53

[101] E. L. Romero, M. Strum, and W. J. Chau, "Comparing two testbench methods for hierarchical functional verification of a bluetooth baseband adaptor," in *CODES+ISSS* (P. Eles, A. Jantsch, and R. A. Bergamaschi, eds.), pp. 327–332, ACM, 2005. 53

[102] A. Silburt, I. Perryman, J. Bergeron, S. Nichols, M. Dufresne, and G. Ward, "Accelerating concurrent hardware design with behavioural modelling and system simulation," in *DAC*, pp. 528–533, 1995. 53

[103] K. R. G. da Silva, E. U. K. Melcher, G. C. S. de Araujo, and V. A. Pimenta, "An automatic testbench generation tool for a systemC functional verification methodology," 2004. 53

[104] S. Fine, S. Ur, and A. Ziv, "Probabilistic regression suites for functional verification," in *Proceedings of the 41st Annual conference on Design Automation (DAC-04)*, (New York), pp. 49–54, ACM Press, June 7–11 2004. 53

[105] C. Valderrama, F. Nacabal, P. Paulin, and A. Jerraya, "Automatic VHDL-C interface generation for distributed cosimulation: Application to large design examples," *Design Automation for Embedded Systems*, vol. 3, no. 2, pp. 199–217, 1998. 54

[106] M. Oyamada and F. Wagner, "Co-simulation of embedded electronic systems," in *12nd European Simulation Symposium. Hamburgo, Alemanha*, Citeseer, 2000. 54

[107] "Cocentric system studio." http://www.synopsys.com/products/cocentric_studio/cocentric_studio.html, 2010. 54

[108] "Seamless cve." http://www.mentor.com/seamless, 2010. 54

[109] F. Hessel, P. LeMarrec, C. Valderrama, A. Romdhani, and A. Jerraya, "MCI-multilanguage distributed co-simulation tool," in *Distributed and Parallel Embedded Systems: Ifip Wg10. 3/Wg10. 5 International Workshop on Distributed and Parallel Embedded Systems (Dipes' 98), October 5-6, 1998, Schlogbs Eringerfeld, Germany*, p. 191, Springer Netherlands, 1999. 54

[110] C. Hylands, E. Lee, J. Liu, X. Liu, S. Neuendorffer, Y. Xiong, Y. Zhao, and H. Zheng, *Overview of the ptolemy project*. Electronics Research Laboratory, College of Engineering, University of California, 2003. 54

[111] M. Dalpasso, A. Bogliolo, and L. Benini, "Virtual simulation of distributed IP-based designs," *IEEE Design & Test of Computers*, vol. 19, no. 5, pp. 92–104, 2002. 54

[112] A. Fin and F. Fummi, "A web-CAD methodology for IP-core analysis and simulation," in *DAC*, pp. 597–600, 2000. 54

[113] D. M. Rao, V. Chernyakhovsky, and P. A. Wilsey, "Wese: A web-based environment for systems engineering," Mar. 11 2000. 54

[114] B. Mello and F. Wagner, "A distributed co-simulation backbone," *SOC Design Methodologies, Kluwer Academic Publishers*, 2002. 54

[115] IEEE, "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)," 2000. 54

[116] S. Edwards, L. Lavagno, E. A. Lee, and A. Sangiovanni-Vincentelli, "Design of embedded systems: Formal models, validation, and synthesis," *Proc. of the IEEE*, vol. 85, year 1997. 55

[117] S. Kim, I. Hwang, Y. Kim, and S. Kim, "A VLSI CHIP FOR ISOLATED SPEECH RECOGNITION SYSTEM," in *Consumer Electronics, 1996. Digest of Technical Papers., International Conference on*, p. 118, IEEE, 2002. 56

[118] S. Yuanyuan, L. Jia, and L. Runsheng, "Single-chip speech recognition system based on 8051 microcontroller core," *IEEE Transactions on Consumer Electronics*, vol. 47, no. 1, pp. 149–153, 2001. 56

[119] K. Nakamura, Q. Zhu, S. Maruoka, T. Horiyama, S. Kimura, and K. Watanabe, "Speech recognition chip for monosyllables," in *ASP-DAC*, pp. 396–399, ACM, 2001. 56, 86

[120] F. Vargas, R. Fagundes, and D. Junior, "A FPGA-based Viterbi algorithm implementation for speech recognition systems," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, vol. 2, pp. 1217–1220, IEEE, 2001. 56, 57, 86

[121] F. Vargas, R. D. R. Fagundes, and D. B. Jr, "Experimental results of a recovery block scheme to handle noise in speech recognition systems," in *Asian Test Symposium*, pp. 224–229, IEEE Computer Society, 2002. 57

[122] G. Marcus and J. Nolazco-Flores, "An FPGA-based coprocessor for the SPHINX speech recognition system: early experiences," in *Reconfigurable Computing and FPGAs, 2005. ReConFig 2005. International Conference on*, pp. 4–27, IEEE, 2005. 57, 70

[123] "Cmu sphinx group open source speech recognition engines." http://cmusphinx.sourceforge.net/html/cmusphinx.php, 2010. 57, 58

[124] "Carnegie mellon university." http://www.cs.cmu.edu/, 2010. 57

[125] C. Lai, S. Lu, and Q. Zhao, "Performance analysis of speech recognition software," in *Proceedings of the Fifth Workshop on Computer Architecture Evaluation using Commercial Workloads*, Citeseer, 2002. 57

[126] J. W. Schuster, K. Gupta, and R. R. Hoare, "Speech silicon AM: an FPGA-based acoustic modeling pipeline for hidden markov model based speech recognition," in *IPDPS*, IEEE, 2006. 58, 70, 86

[127] "Microblaze processor reference guide. embedded development kit edk 10.1i." http://www.xilinx.com/, 2010. 58, 59

[128] E. Lin, K. Yu, R. Rutenbar, and T. Chen, "Moving speech recognition from software to silicon: the In Silico Vox Project," in *Ninth International Conference on Spoken Language Processing*, Citeseer, 2006. 58, 86

[129] S. Ke, Y. Hou, Z. Huang, and H. Li, "A HMM speech recognition system based on FPGA," in *Image and Signal Processing, 2008. CISP'08. Congress on*, vol. 5, pp. 305–309, IEEE, 2008. 59, 86

[130] A. Acero, N. Bernstein, R. Chambers, Y.-C. Ju, X. Li, J. Odell, P. Nguyen, O. Scholz, and G. Zweig, "Live search for mobile: Web services by voice on the cellphone," in *ICASSP*, pp. 5256–5259, IEEE, 2008. 59

[131] P. Silva, N. Neto, A. Klautau, A. Adami, and I. Trancoso, "Speech recognition for brazilian portuguese using the spoltech and ogi-22 corpora," 2008. 67, 71, 72, 79, 128, 129

[132] G. F. G. Yared and F. Violaro, "Algoritmo para redução do número de parâmetros de modelos hmm utilizados em sistemas de reconhecimento de fala contínua," 2005. 67, 72, 79, 129

[133] C. HOSN, L. A. BAPTISTA, T. IMBIRIBIRA, and A. KLAUTAU, "New resources for brazilian portuguese: results for grapheme-to-phoneme and phone classification," 2006. 67, 71, 72, 79, 128, 129

[134] E. SILVA, L. BAPTISTA, H. FERNANDES, and A. KLAUTAU, "Desenvolvimento de um sistema de reconhecimento automático de voz contínua com grande vocabulário para o português brasileiro," 2005. 67, 79, 129

[135] S. Young, "The htk hidden markov model toolkit: Design and philosophy," tech. rep., Cambridge University Engineering Department, 1994. 69

[136] "Atk - api for htk." Disponível em http://htk.eng.cam.ac.uk/develop/atk.shtml, 2010. 69

[137] N. DESHMUKH, A. GANAPATHIRAJU, J. HAMAKER, J. PICONE, and M. ORDOWSKI, "A public domain speech-to-text system," 1999. 70

[138] V. F. S. de Alencar and A. Alcaim, "Lsf and lpc - derived features for large vocabulary distributed continuous speech recognition in brazilian portuguese," 2008. 71, 72, 128, 129

[139] A. A. BRESOLIN, A. D. D. NETO, and P. J. ALSINA, "Brazilian vowels recognition using a new hierarchical decision structure with wavelet packet and svm," 2007. 71, 128

[140] A. A. BRESOLIN, A. D. D. NETO, and P. J. ALSINA, "Digit recognition using wavelet and svm in brazilian portuguese," 2008. 71, 72, 128, 129

[141] R. J. R. C. et al., "Um conjunto de 1000 frases foneticamente balanceadas para o português brasileiro obtido utilizando a abordagem de algoritmos genéticos," 2005. 72

[142] A. Alcaim, J. A. SOLEWICZ, and J. A. MORAES, "Freqüência de ocorrência dos fones e listas de frases foneticamente balanceadas no português falado no rio de janeiro," *Revista da Sociedade Brasileira de Telecomunicações (SBrT)*, vol. 7, no. 1, pp. 23–41, 1992. 72, 73, 129

[143] C. A. Ynoguti and F. Violaro, "A brazilian portuguese speech database," 2008. 73, 129

[144] "Spoltech brazilian portuguese corpus." http://www.cslu.ogi.edu/corpora/spoltech/, 2010. 73

[145] "Corpus de extractus de textos electrônicos nilc/ folha de são paulo (ceten-folha)." http://acdc.linguateca.pt/cetenfolha/, 2010. 73

[146] "Sphinx knowledge base tool." http://www.speech.cs.cmu.edu/tools/lmtool-new.html, 2010. 77

[147] T. F. Zheng, G. Zhang, and Z. Song, "Comparison of different implementations of MFCC," *J. Comput. Sci. Technol*, vol. 16, no. 6, pp. 582–589, 2001. 86, 104

[148] M. D. Skowronski and J. G. Harris, "Increased mfcc filter bandwidth for noise-robust phoneme recognition," in *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, vol. 1, pp. I–801 –I–804, may 2002. 86, 104

[149] A. Zabidi, W. Mansor, L. Y. Khuan, I. Yassin, and R. Sahak, "Investigation of mel frequency cepstrum coefficients parameters for classification of infant cries with hypothyroidism using mlp classifier," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pp. 1 –4, july 2010. 86, 104

[150] S. Madikeri and H. Murthy, "Mel filter bank energy-based slope feature and its application to speaker recognition," in *Communications (NCC), 2011 National Conference on*, pp. 1 –4, jan. 2011. 86, 104

[151] A. Zabidi, L. Y. Khuan, W. Mansor, I. Yassin, and R. Sahak, "Optimization of mfcc parameters using particle swarm optimization for diagnosis of infant hypothyroidism using multi-layer perceptron," in *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pp. 1417 –1420, 31 2010-sept. 4 2010. 86, 104

[152] E. Bocchieri and D. Blewett, "A decoder for lvcsr based on fixed-point arithmetic," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol. 1, p. I, may 2006. 86

[153] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers," in *Proc. Of the 1998 CM/SIGDA Sixth International Symposium on FPGAs*, pp. 191–200, 1998. 111