UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
CENTRO DE TECNOLOGIA
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ELETRÔNICA

Sistema de Reconhecimento de Palavras Isoladas em Tempo Real

Autor: _____
Marcos Salazar Francisco DRE: 094130148

Orientador: _____
Prof. Fernando Gil Vianna Resende Junior, Ph. D.

Co-Orientador: _____
Prof. Sérgio Lima Netto, Ph. D.

Examinador: _____
Prof. Márcio Nogueira de Souza, D. Sc.

DEL
Dezembro de 1999

À minha namorada Danielle de Paiva P. Lopes,
que sempre soube me apoiar e incentivar.

Agradecimentos

Aos meus pais Mauro e Angela, pelo apoio que têm me dado em todas as etapas da minha vida.

Aos professores Gil e Sérgio, pela orientação recebida no desenvolvimento deste trabalho.

Aos professores Márcio Nogueira e Carlos Espain, que gentilmente cederam os sistemas de reconhecimento de fala criados por seus alunos.

Resumo

Este trabalho consiste em um sistema de reconhecimento de palavras isoladas em tempo real, que foi desenvolvido para uma plataforma Windows 9x. Ele reconhece os dígitos de 0 a 9, mas pode-se facilmente expandi-lo para ser utilizado com mais palavras.

O sistema como um todo é formado por cinco blocos principais: aquisição do som; detecção de início e fim de palavra; extração de coeficientes; quantização vetorial; HMM. Estes blocos são descritos ao longo deste relatório.

Há diversas variáveis envolvidas no processo de reconhecimento, mas existe pouca informação sobre a escolha de seus valores. Neste trabalho, analisa-se a influência destas variáveis no desempenho do sistema: tamanho do *codebook*, número de estados nos modelos HMM, número de coeficientes extraídos, tamanho da superposição, tamanho da janela.

A melhor taxa de acerto obtida foi de 98,75% para uma base de dados com um único locutor e de 86,25% para uma base independente do locutor.

Palavras-chaves:

- Reconhecimento de Voz
- Tempo Real
- HMM
- Análise Cepstral
- Análise LPC

Índice:

1. Introdução 1
2. Sistema de reconhecimento 2
   2.1. Detecção de extremos 4
   2.2. Extração de parâmetros 6
      2.2.1. Análise LPC 6
      2.2.2. Análise Cepstral 9
      2.2.3. Análise em Tempo Real 11
   2.3. Quantização vetorial 13
   2.4. Hidden Markov Models (HMM) 15
      2.3.1. Reconhecimento 17
      2.3.2. Treinamento Baum-Welch 20
      2.3.3. Implementação do treinamento Baum-Welch 22
3. Resultados 23
      2.3.1. Treinamento 23
      2.3.2. Reconhecimento dependente do locutor 25
      2.3.3. Reconhecimento independente do locutor 30
4. Conclusões 35
5. Bibliografia 37
6. Apêndice 1 39

1. Introdução

Com os rápidos avanços tecnológicos das últimas décadas, os computadores já são uma realidade na vida diária das pessoas. Com isso, surge a necessidade de interfaces entre o homem e a máquina de forma mais simples, permitindo o uso de computadores por um número maior de pessoas.

A linguagem oral é a forma de comunicação mais rápida e fácil que o ser humano dispõe. Isto motiva o estudo de sistemas de reconhecimento e síntese de voz. Inúmeros sistemas de reconhecimento já foram desenvolvidos, sendo a grande maioria para a língua inglesa. No entanto, existem poucos em português.

Neste trabalho, é implementado um sistema para o reconhecimento de palavras isoladas em tempo real, no qual são utilizados os modelos escondidos de Markov (*Hidden Markov Models - HMM*). O vocabulário de palavras é reduzido (apenas os dígitos de zero até nove), por isso foram usados modelos representando palavras inteiras (um modelo para cada dígito). A principal motivação para esse projeto é a infinidade de aplicações para um sistema de reconhecimento de voz, como exemplo, em telefones com discagem por voz [8], editores de texto [9-10], ou aplicado a interação com robôs [11].

O objetivo principal do trabalho é obter uma biblioteca de programas para o desenvolvimento de sistemas de reconhecimento de fala em tempo real, analisando as variáveis envolvidas e encontrando os melhores valores para elas.

Em [12-18] são vistos trabalhos abordando o reconhecimento de dígitos. Estes sistemas obtêm taxas de acerto de 90%, 95%, 96.25%, 96.19%, 95%, 91%, 97.77%, respectivamente. Todavia, em [16] e [18] foram testados com para os mesmos locutores que realizaram o treinamento. Em [12] e [13], os resultados são para uma base dependente do locutor. Verificando as bases de dados, observa-se que apenas em [15] foi utilizada uma base ampla contendo 600 falantes.

O sistema desenvolvido neste trabalho é uma aplicação para uma plataforma *Windows 9x*. A interface com o usuário foi feita em *Delphi 3.* As funções que processam o sinal de voz e fazem o reconhecimento estão numa *DLL* e foram desenvolvidas no *Borland C++ 5*. Uma descrição mais detalhada dessas funções encontra-se no Apêndice 1.

Este trabalho teve a colaboração de dois professores: o professor Márcio Nogueira de Souza [7] , e o professor Carlos Espain [4]. O primeiro possuía um sistema de reconhecimento de dígitos isolados em tempo real. O segundo possuía um sistema de reconhecimento de palavras isoladas feito em *Matlab*.

No próximo capítulo, trata-se de uma breve descrição da teoria envolvida no tratamento da fala. No Capítulo 3, são apresentados os testes realizados e uma análise dos seus resultados. As conclusões são mostradas no Capítulo 4.

2. Sistema de reconhecimento

O funcionamento básico desse sistema de reconhecimento pode ser resumido no diagrama em blocos visto na Figura 2.1. O sistema é constituído por quatro etapas: detecção de extremos; extração de parâmetros; quantização vetorial; classificação do sinal.

O bloco de detecção de extremos realiza a aquisição dos dados através da placa de som e a decisão de início e fim de palavra (recorte de palavra). O objetivo é descobrir o trecho do sinal de fala, onde existe uma palavra e entregá-lo ao bloco de extração de parâmetros.

Para realizar a extração de parâmetros, é feita uma segmentação do sinal de fala em pequenos trechos da ordem de 20 ms. Isso é importante para que estes possam ser analisados como sinais estacionários e para possibilitar o processamento em tempo real. Em cada um destes segmentos, é feita a análise LPC (**linear predictive coding**) e posterior conversão de coeficientes LPC para coeficientes cepstrais. No fim desse bloco, cada janela do sinal é formada por parâmetros cepstrais (N), delta cepstrais (N), energia e delta energia.

Depois efetua-se uma quantização vetorial, onde cada conjunto de parâmetros é quantizado pelo seu índice no **codebook**, e a seqüência de índices é utilizada no classificador HMM, para que se possa verificar qual palavra foi dita.

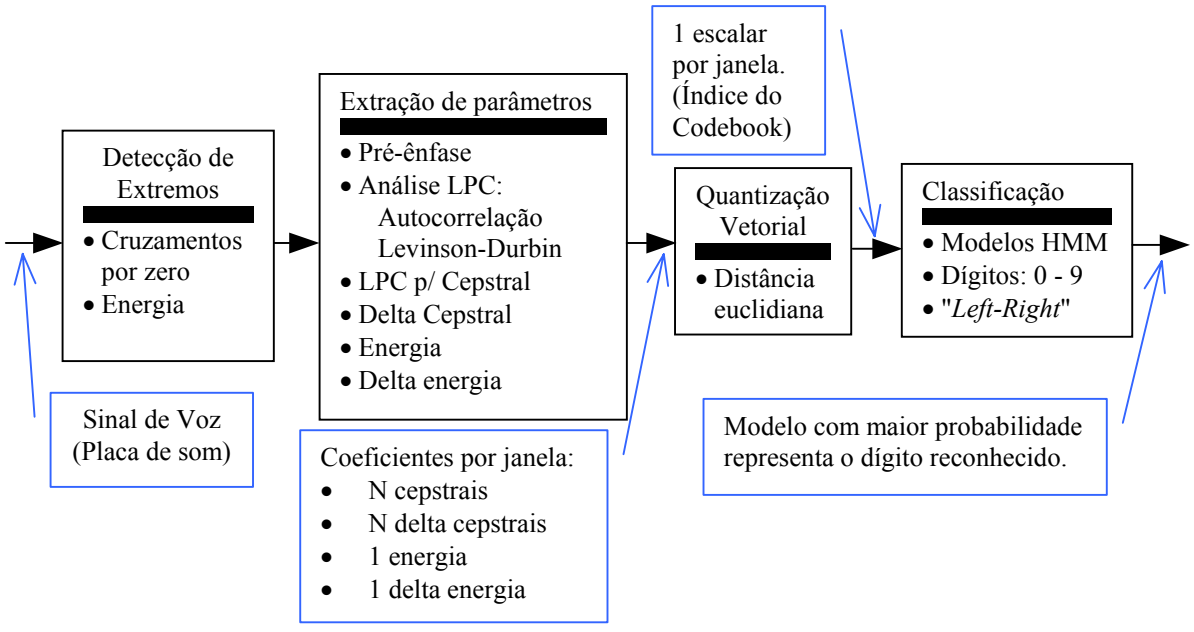


Figura 2.1 : Diagrama em blocos do sistema de reconhecimento implementado neste trabalho.

Para que o sistema funcione, é necessário que sejam treinados: o codebook e os modelos HMM. Na Figura 2.2, pode ser visto um diagrama em blocos para se efetuar o treinamento. Os blocos de detecção de extremos e de extração de parâmetros são os mesmos utilizados antes, porém agora, eles processam um conjunto grande de arquivos de som (**conjunto de treinamento**) ao invés de apenas um. Os blocos seguintes realizam o treinamento do codebook e dos modelos HMM. O primeiro é feito a partir do conjunto de treinamento, utilizando o algoritmo LBG (**Linde-Buzo-Gray**) com **centroid splitting**. O segundo, após todo este conjunto ter sido quantizado.

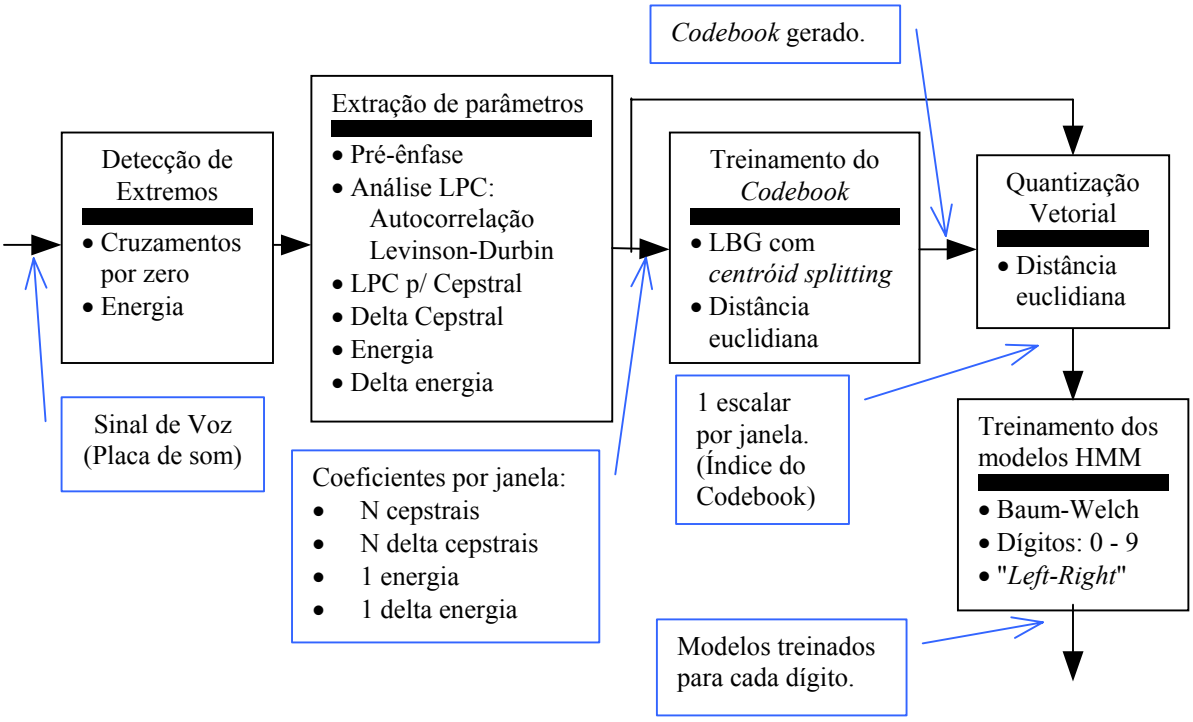


Figura 2.2 : Diagrama em blocos de treinamento do sistema implementado.

2.1. Detecção de extremos

Ao se pronunciar uma palavra, é necessário descobrir os instantes de início e de término dela. Com o bloco de detecção de extremos, consideram-se somente as informações relativas à palavra falada. As amostras anteriores e posteriores do sinal de voz, onde só havia ruído de fundo, são desprezadas.

A presença do bloco de detecção evita que o ruído de fundo seja constantemente processado como se fosse um sinal de fala. Além de ser um processamento desnecessário, poderia gerar resultados indesejáveis no reconhecimento. Mesmo um pequeno erro na detecção dos extremos, já é suficiente para afetar significativamente a taxa de acerto global do sistema [2].

O algoritmo utilizado no recorte se baseia em captar pequenos intervalos de sinal de voz e, através dos cálculos de energia e da taxa de cruzamentos por zero nesses intervalos, identificar o início e o final da palavra.

A energia de um segmento do sinal de fala $s(n)$ com N amostras pode ser calculado pela seguinte equação [1]:

$$E = \sum_{n=0}^{N-1} s^2(n) \qquad (2.1)$$

A taxa de cruzamentos por zero é o número de vezes que uma seqüência muda de sinal. Ela é definida pela equação [1]:

$$Z = \frac{1}{N} \cdot \sum_{n=0}^{N-1} \frac{\left|\operatorname{sgn}\{s(n)\} - \operatorname{sgn}\{s(n-1)\}\right|}{2},$$

onde (2.2)

$$\operatorname{sgn}\{s(n)\} = \begin{cases} +1, & s(n) \geq 0 \\ -1, & s(n) < 0 \end{cases}$$

Os extremos de boa parte das palavras existentes são detectados através da energia, mas existem palavras com trechos **não-vozeados**, principalmente **fricativos**, que possuem uma amplitude de energia bem próxima do ruído de fundo. Como nestes trechos o espectro se concentra nas altas freqüências, a taxa de cruzamento por zero costuma apresentar valores mais altos do que para o ruído de fundo. Desta forma, usa-se esta medida para auxiliar na detecção mais precisa dos extremos do sinal de fala.

Algoritmo: Este sistema funciona pedindo continuamente para que a placa de som grave trechos de 100 ms. A presença de uma palavra é identificada, quando a intensidade do sinal dentro desse bloco ultrapassa um limiar (**Li**). A partir deste ponto, verifica-se trechos anteriores de sinal (em blocos de 25 ms), e efetua-se o cálculo da taxa de cruzamentos por zero e da energia, comparando com seus respectivos limiares (**Lz**, **Le**), até encontrar o ponto de início dela. Ao mesmo tempo, é feita a gravação de um segundo de som, obtendo um vetor. Percorre-se esse vetor de trás para frente em blocos de 25ms, encontrando o fim da palavra através da taxa de cruzamentos por zero e da energia e comparando com os mesmos limiares. Na Figura 2.3, aparece um exemplo para a palavra "sete".

Este algoritmo foi desenvolvido pelo aluno Elias sob orientação do professor Márcio [7]. Apesar de existirem diversos algoritmos de detecção de extremos, este foi escolhido pela facilidade de implementação, por realizar a aquisição de dados em tempo real e por ter resolvido problemas de implementação. Com isso, pode-se dar maior ênfase na implementação nas etapas de extração de parâmetros, quantização vetorial e classificação HMM.

Este método possui algumas deficiências. Ele é frágil em relação a ruídos de fundo e o limite de 1 segundo obriga o usuário a pronunciar no máximo uma palavra por segundo. Outro problema relativo a etapa de detecção é um "sopro" devido ao esvaziamento dos pulmões. Isso é muito comum com palavras terminadas com o fonema /o/. Há também um clique que acontece após final da palavra. Esse clique é devido ao fechamento dos lábios após o pronunciamento da palavra.

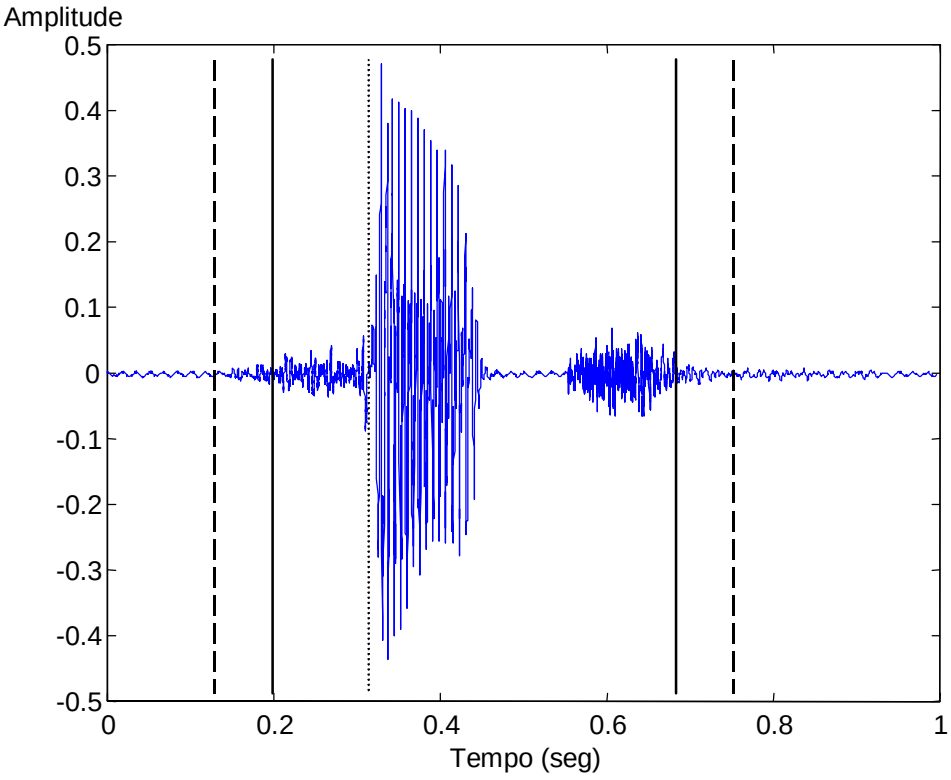


Figure 2.3 : Exemplo de detecção da palavra "sete". O tracejado fino, tracejado longo e a linha contínua correspondem ao recorte feito com os limiares Li, Lz e Le, respectivamente.

2.2. Extração de parâmetros

A partir de um sinal de voz, são encontrados os parâmetros de cada janela do sinal de voz. Esses parâmetros são: a energia da janela; o delta energia da janela; **p** coeficientes cepstrais da janela; e **p** delta cepstrais da janela; num total de **(2p + 2)** parâmetros.

Os parâmetros extraídos por este bloco representam as amostras do sinal com um número menor de coeficientes. Se, por exemplo, usarmos uma freqüência de amostragem de 11,025 KHz e um tamanho de janela de 20 ms, cada segmento teria 220 amostras. Mas se for feita a extração de parâmetros, podemos ter menos que 40 coeficientes por segmento. Com uma menor quantidade de valores, diminui-se o trabalho computacional e agiliza-se o processamento nas etapas posteriores. Além disso, os parâmetros extraídos são uma boa representação da informação acústica.

2.2.1. Análise LPC

A idéia básica da análise LPC consiste em uma amostra do sinal de fala ser modelada por uma combinação linear de suas **p** amostras passadas, dada por:

$$s(n) \approx a_1 s(n-1) + a_2 s(n-2) + ... + a_p s(n-p), \qquad (2.3)$$

onde os coeficientes $a_1, a_2,...,a_p$ são recalculados para cada janela do sinal, pois, em pequenos trechos, o sinal pode ser assumido como sendo estacionário.

A equação anterior pode ser convertida em uma igualdade, incluindo um termo de excitação do sinal, $Gu(n)$, onde $u(n)$ é a excitação normalizada e G é o seu ganho.

$$s(n) = Gu(n) + \sum_{i=1}^{p} a_i s(n-i) \qquad (2.4)$$

Isso nos leva a uma função de transferência do trato vocal como é mostrado abaixo:

$$H(z) = \frac{S(z)}{U(z)} = \frac{G}{1 - \sum_{i=1}^{p} a_i z^{-i}} \qquad (2.5)$$

Os coeficientes $a_1, a_2,...,a_p$ são os parâmetros LPC do sinal. Eles são calculados, considerando-se um filtro de predição linear como mostrado em (2.6) e o seu erro de predição é dado por (2.7).

$$\tilde{s}(n) = \sum_{i=1}^{p} a_i s(n-i) \tag{2.6}$$

$$e(n) = s(n) - \tilde{s}(n) = s(n) - \sum_{i=1}^{p} a_i s(n-i) \tag{2.7}$$

Os coeficientes são escolhidos a fim de minimizar uma função do erro de predição. Para isso, dentro de uma janela de sinal de tamanho N, o erro médio quadrático definido em (2.8) deve ser derivado em função de cada coeficiente a_i e igualado a zero.

$$MSE: E_l = \sum_{n=0}^{N-1} (e(n))^2 = \sum_{n=0}^{N-1} \left(s(n) - \sum_{i=1}^{p} a_i s(n-i) \right)^2 \tag{2.8}$$

$$\frac{\partial E_l}{\partial a_i} = 0 \quad , \quad i = 1,2,...,p \tag{2.9}$$

onde*: l é o número de segmentos.*

Obtendo:

$$\sum_{n=0}^{N-1} s(n-i)s(n) = \sum_{k=1}^{p} a_k \left(\sum_{n=0}^{N-1} s(n-i)s(n-k) \right) \quad , \quad i = 1,2,..., p \tag{2.10}$$

Esta equação pode ser reescrita, usando a definição de covariância dada por:

$$\varphi_l(i,k) = \sum_{n=0}^{N-1} s(n-i)s(n-k)\,, \tag{2.11}$$

e então:

$$\sum_{k=1}^{p} a_k \varphi_l(i,k) = \varphi_l(i,0) \quad , \quad i = 1,2,..., p \tag{2.12}$$

Este sistema de equações é resolvido de forma matricial em (2.13). Esta matriz pode ser decomposta em duas outras, uma triangular superior e outra triangular inferior. Depois disso, utiliza-se o método da retro substituição [3]. Este é chamado de **método da covariância** [1-3].

$$\begin{bmatrix} \varphi_l(1,1) & \varphi_l(1,2) & \varphi_l(1,3) & \Lambda & \varphi_l(1,p) \\ \varphi_l(2,1) & \varphi_l(2,2) & \varphi_l(2,3) & \Lambda & \varphi_l(2,p) \\ \varphi_l(3,1) & \varphi_l(3,2) & \varphi_l(3,3) & \Lambda & \varphi_l(3,p) \\ \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{O} & \mathrm{M} \\ \varphi_l(p,1) & \varphi_l(p,2) & \varphi_l(p,3) & \Lambda & \varphi_l(p,p) \end{bmatrix} \cdot \begin{bmatrix} a(1) \\ a(2) \\ a(3) \\ \mathrm{M} \\ a(p) \end{bmatrix} = \begin{bmatrix} \varphi(1,0) \\ \varphi(2,0) \\ \varphi(3,0) \\ \mathrm{M} \\ \varphi(p,0) \end{bmatrix} \tag{2.13}$$

$$\text{onde}: \varphi_l(i,k) = \varphi_l(k,i)$$

Com uma simples substituição de variáveis, (2.11) pode ser reescrita como :

$$\varphi_1(i,k) = \sum_{n=-i}^{N-1-i} s(n)s(n+i-k) = \sum_{n=-k}^{N-1-k} s(n)s(n+k-i) \qquad (2.14)$$

Como o sinal é processado em janelas de duração finita ($0 \le n \le N\text{-}1$), os limites do somatório podem ser mudados.

$$\varphi_1(i,k) = \sum_{n=0}^{N-1-(i-k)} s(n)s(n+i-k) = \sum_{n=0}^{N-1-(k-i)} s(n)s(n+k-i) \equiv r(|i-k|) \qquad (2.15)$$

Percebe-se que a covariância pode ser substituída pela autocorrelação do sinal. Neste caso, a equação (2.12) torna-se:

$$\sum_{k=1}^{p} a_k r(|i-k|) = r(i) \quad , \quad i = 1,2,...,p \qquad (2.16)$$

Este é chamado de **método da autocorrelação** e foi utilizado neste trabalho. O sistema de equações pode ser visto na sua forma matricial em (2.17). Como a matriz é do tipo **Toeplitz** [1-3], o melhor método para resolvê-la é utilizar o algoritmo de **Levinson-Durbin** [1-3], dado a seguir:

$$\begin{bmatrix} r(0) & r(1) & r(2) & \Lambda & r(p-1) \\ r(1) & r(0) & r(1) & \Lambda & r(p-2) \\ r(2) & r(1) & r(0) & \Lambda & r(p-3) \\ \text{M} & \text{M} & \text{M} & \text{O} & \text{M} \\ r(p-1) & r(p-2) & r(p-3) & \Lambda & r(0) \end{bmatrix} \begin{bmatrix} a(1) \\ a(2) \\ a(3) \\ \text{M} \\ a(p) \end{bmatrix} = \begin{bmatrix} r(1) \\ r(2) \\ r(3) \\ \text{M} \\ r(p) \end{bmatrix} \qquad (2.17)$$

Algoritmo de Levison-Durbin :

$$\begin{aligned} &Valores \quad iniciais : E^{(0)} = r(0) \quad ; \quad k_0 = 0 \\ &Iteração : 1 \le i \le p \\ &E^{(i)} = \left(1 - k_{i-1}^2\right) E^{(i-1)} \\ &k_i = \left\{ r(i) - \sum_{j=1}^{L-1} \alpha_j^{(i-1)} r(|i-j|) \right\} / E^{(i)} \\ &\alpha_i^{(i)} = k_i \\ &\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)}, \; 1 \le j < i \\ &Resultado : a_i = LPC \quad coeficients = \alpha_i^{(p)}, \; 1 \le i \le p \end{aligned} \qquad (2.18)$$

2.2.2. Análise Cepstral

O sinal de voz é formado por uma excitação (*e(n)*) convoluída com a resposta ao impulso do modelo do trato vocal (*θ(n)*). No entanto, em aplicações de reconhecimento de voz, é desejável separá-las. A idéia da análise cepstral é levar o sinal para um domínio onde isso seja possível.

$$s(n) = e(n) * \theta(n) \tag{2.19}$$

O domínio cepstral pode ser dividido em dois: cepstrum real (*real cepstrum* - RC) e cepstrum complexo (*complex cepstrum* - CC). A diferença entre os dois consiste em, no caso do RC, descartar a informação sobre a fase do sinal, ou seja, todo sinal é de fase mínima. Já no cepstrum complexo, os coeficientes cepstrais possuem parte real e imaginária, mantendo a informação sobre a fase do sinal. Neste trabalho, abordaremos apenas o RC.

Se for aplicado o operador logarítmico no espectro de Fourier do sinal, o sinal obtido será uma combinação linear entre a excitação e a resposta ao impulso mostradas anteriormente. Ao aplicarmos a Transformada Inversa de Fourier, levaremos o sinal a um novo domínio, que pode ser chamado de domínio cepstral, e à nova "freqüência" de qüefrência. Nesse novo domínio, a excitação se concentra em regiões de alta qüefrência e a resposta ao impulso em regiões de baixa qüefrência, sendo possível separá-las. As equações que descrevem esse processo podem ser vistas abaixo, onde $c_e(n)$ corresponde à excitação e $c_\theta(n)$ corresponde ao modelo do trato vocal.

$$\begin{aligned}
&s(n) = e(n) * \theta(n) \quad \Leftrightarrow \quad S(w) = E(w).\Theta(w) \\
&Aplicando\ logaritmo: \ log(E(w).\Theta(w)) = log(E(w)) + log(\Theta(w)) \\
&c_s(n) = \Im^{-1}\{|log(E(w).\Theta(w))|\} = \Im^{-1}\{|log(E(w))|\} + \Im^{-1}\{|log(\Theta(w))|\} \\
&c_s(n) = c_e(n) + c_\theta(n) \quad , \quad \begin{cases} c_e(n) = \Im^{-1}\{|log(E(w))|\} \\ c_\theta(n) = \Im^{-1}\{|log(\Theta(w))|\} \end{cases}
\end{aligned} \tag{2.20}$$

Uma vez estando no domínio cepstral, basta separar o componente correspondente ao modelo do trato vocal, pois esta é a informação que interessa em um sistema de reconhecimento de voz. Isso é feito através de um processo de filtragem, chamado liftragem. Neste processo, mantêm-se apenas as baixas qüefrências. Para isso, utiliza-se uma janela (liftro) como mostrada na Figura 2.4. O tamanho da janela depende de quantos coeficientes cepstrais se deseja.

A janela utilizada na liftragem está descrita abaixo:

$$l(n) = \begin{cases} 1 + \frac{L}{2} sin\left(\frac{\pi n}{L}\right), & n = 0,1,...,L \\ 0 \quad , \quad \text{qualquer outro n} \end{cases}$$

onde : (2.21)

$L =$ tamanho da janela de liftering

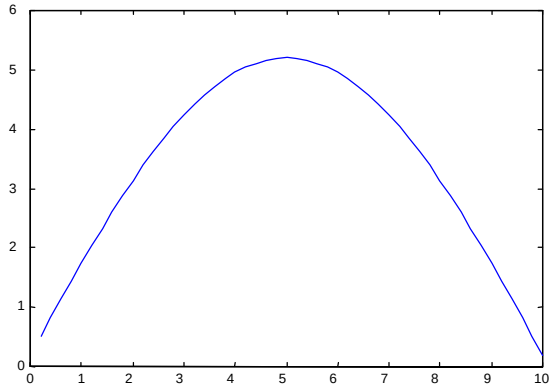


Figura 2.4: Janela utilizada na liftragem com L=10.

O diagrama em blocos mostra todo o processo para a obtenção do cepstrum real, conforme a figura abaixo.

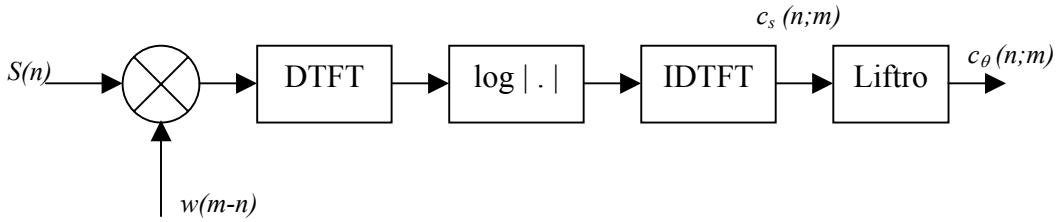


Figura 2.5: Diagrama em blocos para se obter o cepstrum real.

Também é possível obter os coeficientes cepstrais a partir dos coeficientes LPC. Isso costuma ser mais rápido computacionalmente do que o processo descrito anterior e, por isso, foi utilizado neste trabalho. As equações utilizadas para essa conversão são [1-2]:

$$c_0 = ln(G)$$

$$c_m = a_m + \sum_{k=1}^{m-1}\left(\frac{k}{m}\right)c_k a_{m-k} \quad , \; 1 \le m \le p \qquad (2.22)$$

$$c_m = \sum_{k=1}^{m-1}\left(\frac{k}{m}\right)c_k a_{m-k} \quad , \; m \ge p$$

2.2.3. Análise em tempo real

Nesta seção, descreve-se a metodologia utilizada no cálculo dos coeficientes cepstrais, delta cepstrais, energia e delta energia.

1. **Pré ênfase**: É um filtro FIR de primeira ordem, que atenua as componentes de baixa freqüência do sinal. Isto previne a instabilidade numérica[1], devido à matriz de autocorrelação mal condicionada. E também minimiza o efeito da característica dos lábios com o seu zero perto de z=1. A relação entre a saída e a entrada do filtro é dada por:

$$\tilde{s}[n] = s[n] - \mu\, s[n-1] \;, \quad \begin{cases} 0 \le n \le \text{Tamanho do sinal} \\ 0{,}9 \le \mu \le 1.0 \end{cases} \qquad (2.23)$$

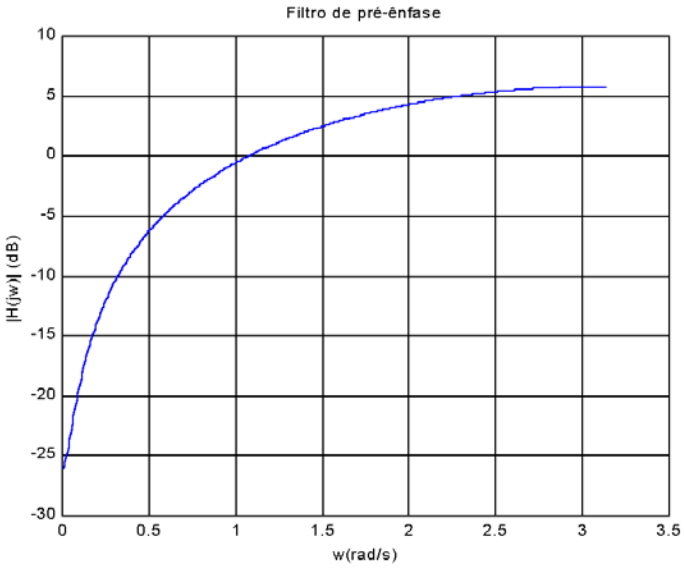


Figura 2.6 : Magnitude do filtro de pré-ênfase com μ=0,95.

2. **Segmentação**: O sinal é processado segmento a segmento, onde cada um possui um tamanho **N**, e a distância do início de um segmento ao seu subseqüente (*overlap* ou superposição) é **O**. Também é utilizada a janela de *Hamming* para diminuir as descontinuidades no início e no fim da janela.

$$\tilde{s}(n) = s(n)\, w(n) \;,$$
$$onde: \; w(n) = 0.54 - 0.46\cos\left(\frac{2n\pi}{N-1}\right), \quad 0 \le n \le N-1 \qquad (2.24)$$

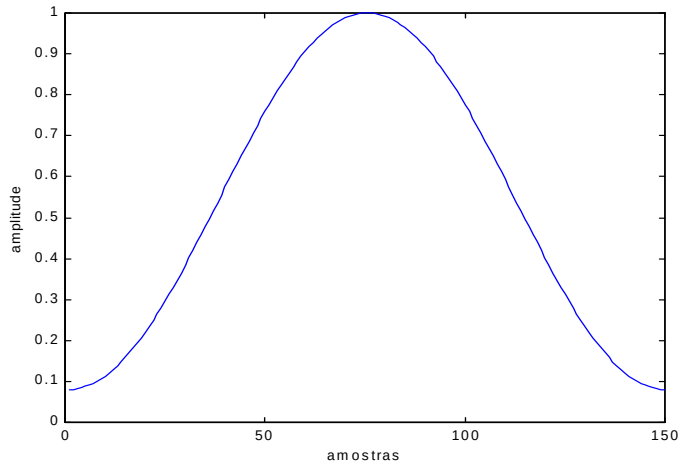


Figura 2.7 : Janela de Hamming com N = 150.

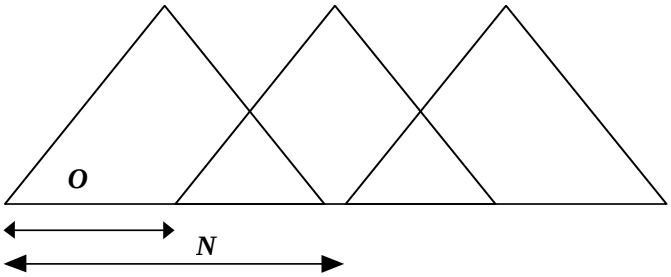


Figura 2.8 : Superposição entres as janelas

3. **Autocorrelação**: Calculam-se *(p +1)* coeficientes de autocorrelação para cada janela do sinal. O termo *r(0)* corresponde à energia da janela e também é um dos parâmetros extraídos.

$$r(m) = \sum_{n=0}^{N-1-m} \left(s(n)s(n+m) \right) \quad , \quad 0 \le m \le p \tag{2.25}$$

4. **Levinson-Durbin:** Os p coeficientes LPC de cada janela são calculados, utilizando o algoritmo de Levinson-Durbin, que foi descrito na seção (2.2.1).

5. **Conversão para coeficientes cepstrais**: Para cada janela do sinal, calculam-se p coeficientes cepstrais a partir dos coeficientes LPC. O primeiro coeficiente cepstral foi desprezado, pois é muito correlacionado com a energia [2]. Como o número de coeficientes cepstrais é o mesmo que o de coeficientes LPC, o cálculo se resume na equação mostrada abaixo [2]:

$$c_m = a_m + \sum_{k=1}^{m-1} \left(\frac{k}{m} \right) c_k a_{m-k} \quad , \; 1 \le m \le p \tag{2.26}$$

6. **Liftragem**: Os cepstrais de cada bloco são normalizados pelo seno conforme abaixo. Isso se deve à sensibilidade dos coeficientes de baixa ordem e de alta ordem, ao espectro do sinal e ao ruído, respectivamente [2].

$$w_m = \left[1 + \frac{p}{2} sin\left(\frac{m\pi}{p} \right) \right] , \ 1 \le m \le p \qquad (2.27)$$

7. **Delta cepstral e delta energia**: É uma medida da variação dos coeficientes cepstrais entre segmentos de sinal adjacentes [2]. O número de janelas utilizadas no cálculo é **(2K+1)** e μ é uma constante de normalização. Neste trabalho, foram usados μ = 1/6 e K = 3 [2]. O delta energia é calculado da mesma forma, utilizando a informação de energia de janelas adjacentes.

$$\frac{\partial c_m(t)}{\partial t} = \Delta c_m(t) \approx \mu \sum_{k=-K}^{K} k c_m(t+k) \qquad (2.28)$$

2.3. Quantização vetorial

Os parâmetros extraídos na etapa anterior podem assumir uma infinidade de valores possíveis. Porém, como veremos na Seção 2.4, são utilizados modelos HMM discretos. Então, necessita-se fazer uma quantização destes valores. Nesta seção, é abordado um método de quantização chamado de quantização vetorial.

Neste método, após a extração de parâmetros de cada janela do sinal, é feita uma comparação entre esses parâmetros e uma lista de parâmetros chamada de codebook. Essa comparação visa encontrar o índice da melhor representação desses parâmetros dentro do codebook, e utiliza-se esse índice para representar a janela do sinal em questão. Esse processo de busca no codebook também é chamado de **quantização vetorial**. A medida de distância utilizada para realizar esta comparação é a distância euclidiana:

$$dist(\hat{x}, \hat{y}) = \sqrt{\sum_{n=1}^{N} (x(n) - y(n))^2} \qquad (2.29)$$

onde: x e y são vetores de dimensão N.

Mas, antes de se utilizar a quantização vetorial, um codebook deve ser gerado a partir de uma lista de parâmetros (conjunto de treinamento). Esta lista deve ser grande o suficiente, de modo a representar com um erro pequeno todo o universo de sinais possíveis. Cada item dessa lista é um conjunto de parâmetros, que pode ser considerado como um vetor de um espaço vetorial de todos os conjuntos de parâmetros possíveis. Obviamente, para se obter o melhor codebook para uma determinada aplicação de reconhecimento de voz, ele deve ser gerado

utilizando-se como conjunto de treinamento amostras de todas as palavras com que o sistema irá trabalhar.

LBG com *centroid splitting* [2] [6]

Inicialização: O primeiro codebook é formado apenas por um único vetor. Este é a média (centróide) de todo o conjunto de treinamento.

Splitting: Dobra-se o tamanho do codebook dividindo cada centróide (y_n) em dois outros. Isso é feito de acordo com a regra: $y_n^+ = y_n(1+\varepsilon)$; $y_n^- = y_n(1-\varepsilon)$. Onde *n* varia de 1 até o tamanho atual do codebook e ε é o parâmetro de splitting (tipicamente ε é escolhido na faixa $0{,}01 \leq \varepsilon \leq 0{,}05$ [2]).

Iteração:
1. Dividem-se todos os vetores do conjunto de treinamento em grupos, utilizando o codebook atual.
2. Calculam-se os centróides de todos os grupos, e atualiza-se o codebook com estes centróides.
3. Se a variação na função custo e/ou o seu tamanho forem pequenos, pare. Caso contrário volte ao primeiro passo. A função custo utilizada neste trabalho é uma média de todas as distâncias entre cada vetor e seu respectivo centróide, como visto abaixo. A cada iteração, esse valor diminui ou se mantém constante [6].

$$D = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{L_i}\sum_{l=1}^{Li} dist(x_i(l), y_i) \qquad (2.30)$$

onde:
M = número de partições.
L_i = número de vetores da partição i.
$x_i(l)$ = vetor número l da partição i.
y_i = centróide da partição i.

Condição de término: Após o término das iterações, deve-se verificar se o tamanho do codebook atingiu o desejado. Caso não tenha sido alcançado, realiza-se um novo splitting e recomeçam-se as iterações.

2.4. Modelos escondidos de Markov

Considere um processo aleatório que pode ser descrito, em qualquer instante de tempo, como um conjunto de **N** estados distintos ($S_1, S_2, \ldots, S_N$). Em instantes de tempo regularmente espaçados, este sistema muda de estado ou não, de acordo com um conjunto de probabilidades, associadas às transições entre estados.

Para um processo markoviano de 1ª ordem, a probabilidade do estado atual (no instante t da seqüência) ser o estado S_j depende apenas do estado S_i presente no instante imediatamente anterior ($t-1$), ou seja:

$$P\left[q_t = S_j \middle| q_{t-1} = S_i\ , q_{t-2} = S_k\ , \ldots\right] = P\left[q_t = S_j \middle| q_{t-1} = S_i\right] \tag{2.31}$$

Considerando que esta probabilidade é independente do tempo, ou seja, se mantém constante para qualquer instante t, pode-se definir a matriz de transição de estados da forma:

$$A = \left\{a_{ij}\right\}_{N\,x\,N} = \left\{P\left[q_t = S_j \middle|\ q_{t-1} = S_i\right]\right\}_{N\,x\,N}, \quad 1 \le i, j \le N \tag{2.32}$$

Nesta matriz, cada linha i e coluna j correspondem a probabilidade de transição do estado S_i para o estado S_j. Sabendo que em cada instante t ocorre uma transição de estados (o próximo estado pode ser o mesmo estado), verifica-se que a matriz de transição de estados satisfaz as seguintes condições:

$$\begin{aligned} &a_{ij} \ge 0 \\ &\sum_{j=1}^{N} a_{ij} = 1 \end{aligned} \tag{2.33}$$

Se esta matriz de estados for utilizada para modelar um processo estocástico, cada estado corresponderá a uma saída observável do sistema. Este processo estatístico é chamado de modelo observável de Markov. Porém, este modelo é muito restrito para ser aplicado a problemas de reconhecimento de voz.

Nestas aplicações, o sinal de voz pode ser representado como dois processos estocásticos dependentes entre si. Um processo é a variação das propriedades aleatórias do sinal ao longo do tempo, e o outro processo é relativo aos parâmetros que modelam o sinal de voz em um determinado instante. Estes parâmetros podem ser considerados como uma variável aleatória e podem ser observados fisicamente. Um modelo capaz de descrever este tipo de sistema é chamado de modelo oculto de Markov (hidden Markov models - HMM) [1] [2] [5].

Para melhor explicar este tipo de modelo, supõe-se o seguinte cenário: existe uma sala com uma cortina e, no lado oculto pela cortina, uma pessoa está realizando um experimento de lançamento de dados. A única informação fornecida é o resultado do lançamento do dado. Neste caso, o processo observável é o resultado do lançamento do dado e o processo oculto é a escolha aleatória do dado que será utilizado no lançamento.

Então, gera-se uma seqüência observada e deseja-se construir um modelo HMM que a explique. A primeira parte do problema é decidir quantos estados serão utilizados no modelo, uma vez que não se sabe o número de dados existentes.

Assume-se, então, que existem **N** dados e cada dado possuindo **M** faces. Denotando-se todos os símbolos possíveis como sendo :

$$O = \{o_1, o_2, \ldots, o_M\}, \tag{2.34}$$

Pode-se definir a matriz de geração de símbolos :

$$B = \{b_j(k)\}_{N\,x\,M} = \left\{P\left[o_k em\, t \,\middle|\, q_t = S_j\right]\right\}_{N\,x\,M}, \begin{cases} 1 \le j \le N \\ 1 \le k \le M \end{cases} \tag{2.35}$$

Nesta matriz, cada linha j corresponde a um vetor com a probabilidade de se observar cada um dos **M** símbolos em um determinado estado j. Sabendo disso, verifica-se que esta matriz satisfaz as seguintes condições:

$$\begin{aligned} & b_j(k) \ge 0 \\ & \sum_{k=1}^{M} b_j(k) = 1 \end{aligned} \tag{2.36}$$

Também é necessário saber a probabilidade de uma determinada seqüência iniciar em um determinado estado S_i. Isto é definido como :

$$\pi_i = P[q_1 = S_i], \quad 1 \le i \le N \tag{2.37}$$

Uma vez escolhidos os valores de **N** e **M**, um modelo HMM pode ser especificado através das três matrizes de probabilidades A, B e π. Por conveniência, ao longo deste trabalho, será usada a notação $\lambda=(A,B,\pi)$ para designar o conjunto completo de parâmetros do modelo.

2.4.1. Reconhecimento

Em um sistema de reconhecimento, existe um modelo para cada informação que se deseja reconhecer, por exemplo, modelos de palavras isoladas. Dada uma seqüência de observações **O** provenientes de uma palavra desconhecida, calcula-se a probabilidade de cada modelo gerar a observação ($P[\lambda|O]$). Obviamente, o modelo que apresentar a maior probabilidade, será escolhido como o modelo correspondente à palavra desconhecida.

De acordo com o teorema de Bayes :

$$P[\lambda|O]P[O] = P[O|\lambda]P[\lambda]$$
$$P[\lambda|O] = \frac{P[O|\lambda]P[\lambda]}{P[O]} \qquad (2.38)$$

Para uma dada observação, a probabilidade de sua ocorrência é constante e independe do modelo. E a probabilidade $P[\lambda]$ é a de estarmos a observar uma dada palavra. Se for admitido que todas as palavras têm igual probabilidade de estarem sendo observadas (isto pode ser válido para um vocabulário pequeno, como o usado neste trabalho), então, esta probabilidade é constante. Nestas condições, maximizar $P[\lambda|O]$ é o mesmo que maximizar $P[O|\lambda]$. Então, basta maximizar a probabilidade de gerar uma determinada seqüência de observações dado um modelo.

Há várias maneiras de se obter esta probabilidade, apenas alguns deles serão descritos neste trabalho.

Procedimento Forward : Considere a probabilidade $\alpha_t(i)$ definida como :

$$\alpha_t(i) = P[o_1 o_2 \ldots o_t, q_t = S_i | \lambda]. \qquad (2.39)$$

Esta é a probabilidade de haver uma seqüência parcial de observações até o instante t terminando no estado S_i, dado um modelo. Então a probabilidade $P[O|\lambda]$ pode ser calculada como se segue :

$$\begin{aligned} &Inicialização : \alpha_1(i) = \pi_i b_i(o_1), \quad 1 \le i \le N \\ &Iteração : \alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}), \quad \begin{cases} 1 \le t \le T-1 \\ 1 \le j \le N \end{cases} \\ &Finalização : P[O|\lambda] = \sum_{i=1}^{N} \alpha_T(i) \quad , \quad T = número\ de\ observações \end{aligned} \qquad (2.40)$$

Procedimento Backward : Considere a probabilidade $\beta_t(i)$ definida como :

$$\beta_t(i) = P\left[o_{t+1} o_{t+2} \ \ldots \ o_T, q_t = S_i \middle| \lambda\right]. \qquad (2.41)$$

Esta é a probabilidade de haver uma seqüência parcial de observações desde o instante $t+1$ até o final (T), estando no estado S_i (no instante t), dado um modelo. Então a probabilidade $P[O|\lambda]$ pode ser calculada como se segue:

$$\begin{aligned}
&\textit{Inicialização} : \begin{cases} \beta_{T+1}(i) = 1 \ , \ \textit{Se } S_i \textit{ é um estado inicial válido} \\ \beta_{T+1}(i) = 0 \ , \ \textit{Caso contrário} \end{cases} \\
&\textit{Iteração} : \beta_t(i) = \left[\sum_{j=1}^{N} \beta_{t+1}(j) a_{ij} b_j(o_{t+1})\right], \quad \begin{cases} t = T-1, T-2, \ldots, 1 \\ 1 \le i \le N \end{cases} \\
&\textit{Finalização} : P\left[O \middle| \lambda\right] = \sum_{i=1}^{N} \pi_i b_i(o_1) \beta_1(i) \quad , \quad T = \textit{número de observações}
\end{aligned} \qquad (2.42)$$

Procedimento de Viterbi : Os procedimentos anteriores calculam a probabilidade $P[O|\lambda]$ para todas as seqüências de estados possíveis. Neste método, calcula-se a probabilidade a partir da seqüência de estados mais provável. Para isso, precisa-se definir a variável :

$$\delta_t(i) = \max_{q_1 q_2 \Lambda \, q_{t-1}} P\left[q_1 q_2 \Lambda \ q_{t-1}, q_t = S_i, o_1 o_2 \Lambda \ o_t \middle| \lambda\right]. \qquad (2.43)$$

Esta é a maior probabilidade calculada dentre todos os caminhos até o instante t, gerando as primeiras t observações e terminando no estado S_i.

Este algoritmo também retorna a melhor seqüência de estados para uma dada observação. Isso é útil para avaliar o número de estados escolhidos para o modelo.

$$\begin{aligned}
&\textit{Inicialização} : \begin{cases} \delta_1(i) = \pi_i b_i(o_1) \\ \psi_1(i) = 0 \end{cases}, \quad 1 \le i \le N \\
&\textit{Iteração} : \begin{cases} \delta_t(j) = \max\limits_{1 \le i \le N}\left[\delta_{t-1}(i) a_{ij}\right] b_j(o_t) \\ \psi_t(j) = \arg \max\limits_{1 \le i \le N}\left[\delta_{t-1}(i) a_{ij}\right] \end{cases}, \quad \begin{cases} 2 \le t \le T \\ 1 \le j \le N \end{cases} \\
&\textit{Finalização} : \begin{cases} P = \max\limits_{1 \le i \le N}\left[\delta_T(i)\right] \\ q_T = \arg \max\limits_{1 \le i \le N}\left[\delta_T(i)\right] \end{cases} \\
&\textit{Seqüência de estados}(\textit{Backtracking}) : \begin{cases} q_t = \psi_{t+1}(q_{t+1}) \\ t = T-1, T-2, \Lambda \ , 1 \end{cases}
\end{aligned} \qquad (2.44)$$

Procedimento de Viterbi alternativo: Aplicando-se o operador logarítmico, o algoritmo de Viterbi, mostrado na seção anterior, pode ser implementado sem a necessidade de realizar multiplicações. Desta forma, temos :

$$
\begin{aligned}
&Preprocessamento: \begin{cases} \overline{\pi}_i = log(\pi_i) & , \quad 1 \le i \le N \\ \overline{b}_i(o_t) = log\left[b_i(o_t)\right], & 1 \le i \le N \ , \ 1 \le t \le T \\ \overline{a}_{ij} = log(a_{ij}) & , \quad 1 \le i, j \le N \end{cases} \\
&Inicialização: \begin{cases} \overline{\delta}_1(i) = log(\delta_1(i)) = \overline{\pi}_i + \overline{b}_i(o_1) \\ \psi_1(i) = 0 \end{cases}, \quad 1 \le i \le N \\
&Iteração: \begin{cases} \overline{\delta}_t(j) = log(\delta_t(j)) = \max\limits_{1 \le i \le N}\left[\overline{\delta}_{t-1}(i) + \overline{a}_{ij}\right] + \overline{b}_j(o_t) \\ \psi_t(j) = arg \max\limits_{1 \le i \le N}\left[\overline{\delta}_{t-1}(i) + \overline{a}_{ij}\right] \end{cases}, \quad \begin{cases} 2 \le t \le T \\ 1 \le j \le N \end{cases} \\
&Finalização: \begin{cases} P = \max\limits_{1 \le i \le N}\left[\overline{\delta}_T(i)\right] \\ q_T = arg \max\limits_{1 \le i \le N}\left[\overline{\delta}_T(i)\right] \end{cases} \\
&Seqüência\ de\ estados(Backtracking): \begin{cases} q_t = \psi_{t+1}(q_{t+1}) \\ t = T-1, T-2, \Lambda \ , 1 \end{cases}
\end{aligned}
\tag{2.45}
$$

Comparando-se os custos computacionais (**O[ ]**) dos quatro métodos na tabela abaixo, percebe-se claramente que o último método é o mais eficiente, pois não necessita de multiplicações. Por isso, optou-se por utilizar este método neste trabalho.

Tabela 2.1 : Comparação dos custos computacionais entre os métodos de reconhecimento [2].

	Adições	**Multiplicações**	**Comparações**
Forward	$O[N^2M]$	$O[N^2M]$	-
Backward	$O[N^2M]$	$O[N^2M]$	-
Viterbi	-	$O[N^2M]$	$O[N^2M]$
Viterbi alternativo	$O[N^2M]$	-	$O[N^2M]$

2.4.2. Treinamento Baum-Welch

O objetivo do treinamento é obter a matriz de probabilidades de transição de estados (A), a probabilidade de iniciar em um determinado estado (π_i) e, para cada estado, um vetor de probabilidades de emissão de cada observação (B).

O método utilizado é simples, mas é necessário que sejam definidas algumas probabilidades para entendê-lo completamente. Essas definições são mostradas abaixo :

- Dado um modelo e uma seqüência de observações, a probabilidade de existir uma transição do estado S_i para o estado S_j no instante t é:

$$\xi_t(i,j) = P\left[q_{t+1} = S_j, q_t = S_i \middle| O, \lambda\right] = \frac{P\left[q_{t+1} = S_j, q_t = S_i, O \middle| \lambda\right]}{P\left[O \middle| \lambda\right]}$$
$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P\left[O \middle| \lambda\right]} \qquad (2.46)$$

- Probabilidade do modelo se encontrar no instante t no estado S_i dada uma observação e um modelo.

$$\begin{cases} \nu_t(i) = \sum_{j=1}^{N} \xi_t(i,j) & , \quad t < T \\ \nu_T(i) = \dfrac{\alpha_T(i)}{P[O \mid \lambda]} & , \quad t = T \end{cases} \qquad (2.47a)$$

ou

$$\nu_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P\left[O \middle| \lambda\right]} \quad , \quad 1 \le t \le T \qquad (2.47b)$$

- Probabilidade do modelo se encontrar no estado S_i no instante t e produzir a saída v_k.

$$\delta_t(i,k) = \begin{cases} \nu_t(i), O_t = v_k \\ 0 \quad , O_t \neq v_k \end{cases} \qquad (2.48)$$

Com estas definições, é possível reestimar as matrizes de probabilidade do modelo HMM. Mas, na prática, para realizar um sistema de reconhecimento, é necessário treinar o modelo para um grande número de amostras. Supondo que temos **H** amostras desta classe, um conjunto de fórmulas para reestimar as probabilidades são mostradas nas equações (2.49).

$$\bar{a}_{ij} = \frac{\textit{número esperado de transições do estado } S_i \textit{ para o estado } S_j}{\textit{número esperado de transições a partir do estado } S_i \textit{ para um estado qualquer}}$$

$$\bar{a}_{ij} = \frac{\sum_{h=1}^{H}\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{h=1}^{H}\sum_{j=1}^{N}\sum_{t=1}^{T-1}\xi_t(i,j)} = \frac{\sum_{h=1}^{H}\sum_{t=1}^{T-1}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{h=1}^{H}\sum_{j=1}^{N}\sum_{t=1}^{T-1}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)} \qquad (2.49a)$$

$$\bar{b}_j(k) = \frac{\textit{número esperado de vezes de estar no estado } S_i \textit{ e observar o símbolo } o_k}{\textit{número esperado de vezes de estar no estado } S_i}$$

$$\bar{b}_j(k) = \frac{\sum_{h=1}^{H}\sum_{t=1}^{T}\delta_t(i,k)}{\sum_{h=1}^{H}\sum_{t=1}^{T}\nu_t(i)} = \frac{\sum_{h=1}^{H}\sum_{t=1}^{T}\alpha_t(i)\beta_t(i)}{\sum_{h=1}^{H}\sum_{t=1}^{T}\alpha_t(i)\beta_t(i)} \quad \textit{Sendo } O_t = v_k \qquad (2.49b)$$

$$\bar{\pi}_i = \textit{número esperado de vezes de se começar no estado } S_i = \nu_1(i) \qquad (2.49c)$$

onde: H é o número de amostras desta classe.

Algoritmo de Baum-Welch

Inicialização: As matrizes A, B e π de cada modelo são inicializadas com valores aleatórios. Obedecendo as restrições (2.33) e (2.36).

Iteração:

1. Utiliza-se as equações (2.49) para reestimar os modelos. Cada modelo é reestimado após serem aplicadas todas as *H* amostras de sua respectiva classe.
2. Se a variação em A, B e π for menor que um limiar pré estabelecido, para-se a iteração.

Repetição: Repete-se os passos para vários modelos iniciais diferentes para se encontrar o máximo local mais favorável [1].

2.4.3. Implementação do treinamento Baum-Welch

A probabilidade de iniciar em um determinado estado (π_i) não é calculada, pois iremos utilizar, neste trabalho, um modelo HMM do tipo left-right [2,4,5]. Na Figura 2.8, é mostrado um exemplo com 4 estados. Neste tipo de modelo, o estado inicial é sempre o mais a esquerda, e só são permitidas transições para estados a direita do anterior.

Como mencionado anteriormente, o sinal de voz pode ser representado como dois processos estocásticos dependentes entre si. Um processo é a variação das propriedades aleatórias do sinal ao longo do tempo, e o outro processo é relativo aos parâmetros que modelam o sinal de voz em um determinado instante. Estes parâmetros podem ser considerados como uma variável aleatória e podem ser observados fisicamente.

A matriz de transição modela o primeiro processo e o modelo do tipo left-right explora o fato de o tempo ser apenas crescente.

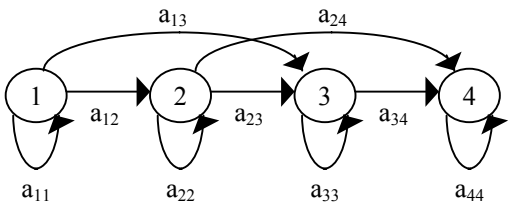


Figura 2.8 : Exemplo de um modelo HMM do tipo *left-right*

3. Resultados

Existem diversos fatores envolvidos num processo de reconhecimento de fala. Porém, na literatura, não há um método geral para a escolha deles. Neste capítulo, são mostrados resultados, variando-se alguns destes parâmetros, e é feita uma análise de sua influência.

Para estes testes, foram utilizados os programas desenvolvidos (Apêndice 1). O ambiente de trabalho foi um computador *Pentium 100MHz*, com *Windows 98* e com *40 Mb de memória RAM*. Devido ao compartilhamento de recursos no Windows, os tempos de processamento mostrados variam ligeiramente (±5%). Com a utilização de computadores mais rápidos, estes tempos devem ser bem menores.

Como mencionado na seção (2.4.2), deve-se repetir o treinamento dos modelos para encontrar o melhor máximo local. Repetiu-se cinco vezes cada um dos testes. Nestas repetições, a taxa média de acerto variou, aproximadamente, de 1% para uma base de dados dependente do locutor e de 3% para uma base independente.

3.1. Treinamento

Antes do sistema ser utilizado, o codebook e os modelos HMM devem ser obtidos. Nesta seção, são vistos os tempos de processamento desse treinamento. Para isso, utiliza-se uma base de dados de 200 arquivos de som. Essa base é a mesma que será usada na próxima seção (base de dados dependente do locutor).

Primeiramente, a extração de parâmetros é feita em diversos arquivos de som, obtendo-se o conjunto de treinamento. Esses parâmetros são: a energia; o delta energia; **N** coeficientes cepstrais; e **N** delta cepstrais; num total de **(2N + 2)**.

O tempo de processamento varia conforme aparece nas tabelas 3.1a, 3.1b e 3.1c. Ele aumenta de acordo com o número de parâmetros (**N**) e a superposição entre as janelas. Porém, ele é maior para um tamanho de janela pequeno, pois, nesse caso, o número total de janelas processadas será maior.

Os próximos passos são: gerar o codebook e utilizá-lo para quantizar o conjunto de treinamento. Esse conjunto quantizado será usado posteriormente no treinamento dos modelos HMM. As tabelas 3.2, 3.3 e 3.4 mostram os tempos gastos em função do número de parâmetros, superposição e tamanho das janelas, respectivamente.

Após isso, é feito o treinamento dos modelos HMM. Nesse processo, o tempo depende do tamanho do codebook, do número de estados e de como foram inicializados os modelos HMM. Sendo mais influenciado por este último fator.

A inicialização dos modelos é feita através de números aleatórios, que influenciam o número de iterações do algoritmo de Baum-Welch. Por isso, o tempo

gasto pode assumir uma ampla faixa de valores. Desde 17 segundos até 125 segundos.

Comparando os tempos de todos os processos, percebe-se que o conjunto de treinamento e os modelos HMM podem ser gerados em um tempo pequeno. Porém, para a maioria de aplicações, deve-se treinar o codebook previamente.

Tabela 3.1 : Tempo gasto na extração de parâmetros do conjunto de treinamento com 200 arquivos de som em função: (a) do número de parâmetros cepstrais com janela de 20 ms e superposição de 66%; (b) da superposição entre as janelas com janela de 20 ms e 14 parâmetros cepstrais; (c) do tamanho das janelas com superposição de 66% e 14 parâmetros cepstrais.

N	8	9	10	11	12	13	14	15	16
T(Seg)	8,92	9,63	10,83	11,43	12,15	12,98	14,12	15,05	16,09

(a)

	0%	33%	50%	66%
T(Seg)	5,13	7,79	9,94	14,12

(b)

	15ms	18ms	20ms	22ms	25ms
T(Seg)	15,24	15,06	14,12	14,10	13,31

(c)

Tabela 3.2 : Tempo gasto na geração do codebook em função do número de parâmetros extraídos. O tamanho da janela usado foi de 20 ms e a superposição é de 66%. O tempo está em (minutos : segundos : centésimos de segundo).

	Tamanho do codebook					
N	32	64	128	256	512	1024
8	1:46:28	3:39:65	8:56:29	12:51:40	16:30:80	24:11:90
9	1:31:89	5:34:22	12:32:76	-	-	-
10	3:03:45	6:03:38	13:35:10	-	-	-
11	3:31:95	5:24:00	9:54:74	-	-	-
12	4:26:66	8:28:77	14:57:48	-	-	-
13	3:27:68	8:18:83	14:06:13	-	-	-
14	4:14:63	10:06:98	17:24:14	-	-	-
15	5:00:22	11:37:48	16:53:15	-	-	-
16	5:25:10	12:53:46	22:49:29	-	-	-

Tabela 3.3 : Tempo gasto na geração do codebook em função da superposição entre as janelas.
O tamanho da janela é de 20 ms e o número de parâmetros é (14+14+2).
O tempo está em (minutos : segundos : centésimos de segundo).

		Tamanho do codebook		
		32	**64**	**128**
Super-posição	**0%**	0:51:19	2:53:12	3:21:58
	33%	2:09:13	2:57:90	6:31:70
	50%	2:56:40	4:56:60	10:41:40
	66%	4:14:63	10:06:98	17:24:14

Tabela 3.4 : Tempo gasto na geração do codebook em função do tamanho das janelas.
A superposição é de 66% e o número de parâmetros é (14+14+2).
O tempo está em (minutos : segundos : centésimos de segundo).

		Tamanho do codebook		
		32	**64**	**128**
Janela	**15ms**	5:52:19	24:15:14	30:46:87
	18ms	4:25:79	8:09:82	16:24:16
	20ms	4:14:63	10:06:98	17:24:14
	22ms	3:51:34	9:22:60	16:56:89
	25ms	2:33:24	6:21:46	11:02:24

3.2. Reconhecimento dependente do interlocutor

Foram utilizadas 20 amostras de cada dígito para realizar o treinamento do codebook e dos modelos HMM. E 40 amostras de cada dígito para todos os testes realizados. Todas estas amostras foram gravadas com uma freqüência de 11,025 KHz e possuem um tamanho entre 180 ms e 800 ms. Como elas foram gravadas por um único interlocutor, o sistema é considerado como dependente do interlocutor.

Os resultados aparecem nas próximas tabelas. Em cada teste é alterada uma das variáveis e as demais são mantidas constantes. A Tabela 3.5 mostra a taxa de acerto em função do tamanho do codebook, mantendo as demais variáveis envolvidas constantes. Na Tabela 3.6, observa-se outro teste, no qual se varia o número de estados. Nas tabelas 3.7, 3.8 e 3.9, são mostrados resultados em função do número de coeficientes cepstrais, tamanho da superposição e da janela, respectivamente.

Todas as tabelas são relativas às 400 amostras de teste. Como a maioria dos testes com o conjunto utilizado no treinamento obteve 100% de acerto, colocou-se uma marcação (do tipo(1)) apenas naqueles com taxa de acerto menor.

Outro fator muito importante a ser considerado é a detecção de extremos. Os sinais, que possuíam um erro visível na detecção, foram "recortados manualmente". Este recorte foi feito através da visualização da forma de onda e da audição de cada sinal.

Em todos os testes, aparece o tempo gasto no reconhecimento de todas as 40 amostras de cada dígito, num total de 400 amostras. Percebeu-se que o tamanho do codebook e o tamanho da superposição são os fatores que mais afetam o tempo de processamento. E também, para 512 e 1024 centróides, o tempo médio gasto para reconhecer um dígito é de 180 ms e 330 ms, respectivamente. Esse tempo é da mesma ordem que o tamanho das palavras, pois a palavra de menor tamanho possui 220 ms.

Vale lembrar que foi utilizado um computador *Pentium 100Mz* com *40Mb de RAM*. Se forem usados computadores de última geração, o tempo de processamento será muito menor, garantindo, assim, a resposta em tempo real do sistema.

Para esta base de dados, a melhor taxa de acerto no reconhecimento foi de 98,75% e os melhores resultados obtidos para os valores são mostrados abaixo:

- **Tamanho da janela**: de 18 ms a 22 ms
- **Superposição**: 66% e 50%
- **Número de parâmetros**: 12 a 14
- **Tamanho do *codebook***: 128
- **Número de estados**: 3 e 4

Os tamanhos 128 e 256 para o codebook apresentaram resultados equivalentes do ponto de vista estatístico, mas o tempo gasto no processamento para 128 é bem inferior, por isso, foi escolhido.

Para 7 ou 8 estados nos modelos HMM, verificou-se que dentre todas as seqüências de estados possíveis, alguns estados não possuíam ocorrências. Isso mostra que o número de estados deve ser menor que 7.

A Tabela 3.10 mostra dígitos reconhecidos em função dos dígitos apresentados. Percebe-se que o maior problema no reconhecimento é para os dígitos "três", "seis" e "sete". Devido ao sotaque do locutor, a região final desses dígitos é muito parecida. Felizmente, o erro nesses dígitos foi muito pequeno para o essa base de dados dependente do locutor.

Tabela 3.5 : Porcentagens de acerto no reconhecimento
em função do tamanho do *Codebook*.

Tam. Cbk	32	64	128	256	512	1024
0	97,5	97,5	100	97,5	92,5	95
1	85	87,5	85	87,5	92,5	95
2	97,5	100	100	100	95	95
3	97,5(1)	95	100	95	95	90
4	90	72,5	97,5	95	95	97,5
5	100	100	100	100	100	100
6	92,5(1)	92,5(1)	95	92,5	95	90
7	87,5(1)	90	95	100	92,5	100
8	100	97,5	100	100	97,5	100
9	77,5	100	95	92,5	85	92,5
T(Seg)	17,3	20,9	28,1	42,2	72,2	132
Média	92,5	93,25	96,75	96	94	95,5

Análise do Tamanho do *Codebook*:
Freq.: 11.025Hz
Janela: 20 ms
Superposição: 66%
Pré-ênfase: 0,95
Núm. Par.: 8+8+2
Núm. Estados: 4

Observações:
100% de acerto testando com o conjunto de treinamento, exceto para:
(1) 95%

Tabela 3.6 : Porcentagens de acerto no reconhecimento
em função do número de estados.

Núm. Est.	3	4	5	6	7(*)	8(*)
0	97,5	100	97,5	100	97,5	100
1	90	85	82,5	77,5	92,5	85
2	100	100	97,5	100	100	100
3	95	100	97,5	92,5	95	95
4	95	97,5	97,5	95	95	85
5	100	100	100	100	100	100
6	95(1)	95	92,5	97,5	95	92,5
7	92,5	95	95	92,5	92,5	97,5
8	100	100	100	100	100	100
9	97,5	95	90	95	92,5	100
T (Seg)	27,6	28,1	28,7	29,5	30,3	31,2
Média	96,25	96,75	95	95	96	95,5

Análise do Número de Estados:
Freq.: 11.025Hz
Janela: 20 ms
Superposição: 66%
Pré-ênfase: 0,95
Núm. Par.: 8+8+2
Tam. Cbk: 128

Observações:
(*) Freqüentemente não há ocorrência de alguns estados.

100% de acerto testando com o conjunto de treinamento, exceto para:
(1) 95%

Tabela 3.7 : Porcentagens de acerto no reconhecimento
em função do número de coeficientes cepstrais.

Análise do Número de Parâmetros (N+N+2):
Freq.: 11.025Hz **Superposição**: 66% **Tam. Cbk**: 128
Janela: 20ms **Pré-ênfase**: 0,95 **Núm. Estados**: 4

N.	8	9	10	11	12	13	14	15	16
0	100	95	97,5	97,5	100	100	100	100	100
1	85	87,5	95	90	100	100	100	100	100
2	100	97,5	97,5	97,5	100	97,5	100	100	100
3	100	100	95	100	95	95	100	97,5	97,5
4	97,5	90	100	100	100	97,5	100	97,5	100
5	100	100	100	100	100	100	100	100	100
6	95	97,5	90	87,5	95	95	92,5	90	90
7	95	90	92,5	90	100	100	97,5	92,5	90
8	100	100	100	100	97,5	100	100	100	100
9	95	95	97,5	95	95	97,5	97,5	100	100
T (Seg)	28,1	30,7	33,2	35,6	37,7	40,4	43,1	45,6	47,9
Média	96,75	95,25	96,5	95,75	98,25	98,25	98,75	97,75	97,75

Tabela 3.8 : Porcentagens de acerto no reconhecimento
em função do tamanho da superposição.

Superposição	66%	50%	33%	0%
0	100	100	97,5	100
1	100	100	97,5	97,5
2	100	97,5	97,5	100
3	100	100	97,5	92,5
4	100	100	97,5	97,5
5	100	100	100	100
6	92,5	97,5	92,5	82,5(1)
7	97,5	97,5	92,5	90
8	100	95	100	100
9	97,5	97,5	100	100
Tempo(Seg)	43,1	30,5	23,9	16,2
Média	98,75	98,5	97,25	96

Análise do Tamanho da Superposição das Janelas:
Freq.: 11.025Hz
Janela: 20 ms
Pré-ênfase: 0,95
Núm. Par.: 14+14+2
Tam. Cbk: 128
Núm. Estados: 4

Observações:
100% de acerto testando com o conjunto de treinamento, exceto para:
(1) 95%

Tabela 3.9 : Porcentagens de acerto no reconhecimento
em função do tamanho das janelas.

Tam. Janelas	15ms	18ms	20ms	22ms	25ms
0	100	100	100	100	100
1	95	100	100	100	100
2	97,5	100	100	97,5	97,5
3	100	97,5	100	92,5	97,5
4	97,5	100	100	100	100
5	100	100	100	100	100
6	92,5	92,5	92,5	97,5	87,5
7	87,5	95	97,5	100	95
8	97,5	97,5	100	97,5	100
9	100	100	97,5	100	100
Tempo(Seg)	55,2	48,3	43,1	40,9	37,1
Média	96,75	98,25	98,75	98,5	97,75

Análise do Tamanho das Janelas:
Freq.: 11.025Hz
Superposição: 66%
Pré-ênfase: 0,95
Núm. Par.: 14+14+2
Tam. Cbk: 128
Núm. Estados: 4

Tabela 3.10 : Tabela de confusão para a melhor taxa de acerto encontrada no teste.

Melhores resultados:
Freq.: 11.025Hz **Pré-ênfase**: 0,95 **Tam. Cbk**: 128
Janela: 20 ms **Núm. Par**.: 14+14+2 **Núm. Estados**: 4
Superposição: 66%

		Dígito reconhecido (acerto médio de 98,75%)										
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Média**
Dígito apresentado	**0**	40										100
	1		40									100
	2			40								100
	3				40							100
	4					40						100
	5						40					100
	6				2			37	1			92,5
	7				1				39			97,5
	8									40		100
	9					1					39	97,5

3.3. Reconhecimento independente do interlocutor

A base de dados utilizada possui 5 amostras de cada dígito para 17 falantes diferentes (13 homens e 4 mulheres). Num total de 850 arquivos de som. Esta base foi cedida pelo professor Márcio [7]. Ela foi dividida em três pedaços:

1. **Treino**: 3 amostras de cada dígito com 9 falantes diferentes (7 homens e 2 mulheres). Num total de 270 arquivos de som.
2. **Teste 1**: 2 amostras de cada dígito com os mesmos falantes. Num total de 180 arquivos de som.
3. **Teste 2**: 5 amostras de cada dígito com 8 falantes diferentes (6 homens e 2 mulheres). Num total de 400 arquivos de som.

Todas estas amostras foram gravadas com uma freqüência de **11,025KHz**. O treinamento do codebook e dos modelos HMM foram feitos com a parte de treino.

Em cada teste é alterada uma das variáveis e as demais são mantidas constantes. A Tabela 3.10 mostra a taxa de acerto em função do tamanho do codebook, mantendo as demais variáveis envolvidas constantes. Na Tabela 3.11, observa-se outro teste, no qual se varia o número de estados. Nas tabelas 3.12 e 3.13, são mostrados resultados em função do número de coeficientes cepstrais e do tamanho da superposição entre as janelas, respectivamente.

Assim como na Seção 3.2, os sinais com erro na detecção foram "recortados manualmente". Este recorte foi feito através da visualização da forma de onda e da audição de cada sinal.

Os três tamanhos para o codebook apresentaram resultados equivalentes do ponto de vista estatístico, mas o tempo gasto no reconhecimento e no treinamento cresce "geometricamente" com o número de centróides. Então, optou-se por utilizar nos próximos testes o valor de 256.

Na Tabela 3.11, observa-se que a taxa de acerto e o tempo de processamento são praticamente constantes. Porém, para 6 estados nos modelos HMM, verificou-se que dentre todas as seqüências de estados possíveis, alguns estados possuíam poucas ou nenhuma ocorrência. Então, nos próximos testes, optou-se por 5 estados.

Em relação a Tabela 3.12, um aumento de 2 coeficientes proporciona um aumento de cerca de 10% no tempo de processamento. Tanto para a base de **treino** quanto para a base de **teste 1**, a taxa de acerto foi a mesma. Porém, o melhor resultado obtido para a base **teste 2** foi de 18, mas o resultado foi bem próximo para 16.

Analisando a superposição entre as janelas, a taxa de acerto média com a base **teste 2** é melhor quando diminui-se a superposição. Entretanto, para a base **teste 1**, ocorre o contrário, o melhor resultado é para 66% de superposição.

Quanto maior a superposição, maior é o tempo de processamento. Analisando esse tempo e a taxa de acerto, verificou-se o melhor resultado para 0%.

As Tabelas 3.15a e 3.15b mostram dígitos reconhecidos em função dos dígitos apresentados. Assim como no caso dependente do locutor, percebe-se que

os maiores problemas no reconhecimento são observados para os dígitos "dois", "três", "seis" e "sete". Devido ao sotaque da maior parte dos locutores, a região final desses dígitos é muito parecida Outro problema é em relação ao dígito "um" ser confundido com o dígito "cinco".

Uma sugestão para resolver esse problema é utilizar um pós-processamento. Usando, por exemplo, a taxa de cruzamento por zero na região inicial dos dígitos, podem-se ser separados o "cinco" do "um", "dois" e "três" do "seis" e "sete", pois essa taxa é maior no início dos dígitos "cinco", "seis" e "sete". E o fato de haver uma pausa antes do fonema /t/ pode ajudar a identificar o dígito 7.

Esse pós-processamento seria útil, mas uma posterior expansão do vocabulário do sistema se tornaria mais complicada.

Tabela 3.11 : Porcentagens de acerto no reconhecimento em função do tamanho do *Codebook*.

Análise do Tamanho do *Codebook*:
Freq.: 11.025Hz **Superposição**: 66% **Núm. Par**.: 14+14+2
Janela: 20ms **Pré-ênfase**: 0,95 **Núm. Estados**: 4

Tam.	128			256			512		
Base	Treino	Tes1	Tes2	Treino	Tes1	Tes2	Treino	Tes1	Tes2
0	100	100	77,5	100	100	92,5	100	94,44	95
1	100	66,66	27,5	100	88,89	40	100	100	47,5
2	100	100	67,5	100	100	77,5	100	100	60
3	100	100	87,5	100	83,33	65	100	83,33	62,5
4	100	100	90	100	100	100	100	100	95
5	100	100	100	100	100	97,5	100	100	90
6	96,3	88,89	82,5	100	77,78	52,5	100	100	70
7	100	100	77,5	100	100	90	100	100	85
8	100	94,44	90	100	100	90	100	100	95
9	96,3	83,33	72,5	100	94,44	77,5	100	100	70
T(Seg)	45,1	28,7	70,4	68,4	43,5	104,4	116	74,1	180
Média	99,26	93,33	77,25	100	94,44	78,25	100	97,78	77

Tabela 3.12 : Porcentagens de acerto no reconhecimento em função do número de estados. Os resultados para 4 estados podem ser vistos nas colunas com tamanho de codebook 256 na Tabela 3.11.

Análise do Número de Estados:
Freq.: 11.025Hz **Superposição**: 66% **Núm. Par**.: 14+14+2
Janela: 20ms **Pré-ênfase**: 0,95 **Tam. Cbk**: 256

Núm.	3			5			6		
Base	Treino	Tes1	Tes2	Treino	Tes1	Tes2	Treino	Tes1	Tes2
0	100	94,44	90	100	100	80	100	100	87,5
1	100	88,89	32,5	100	100	37,5	100	88,89	35
2	100	100	70	100	100	72,5	100	100	60
3	96,3	83,33	62,5	88,89	77,78	62,5	96,3	88,89	70
4	100	100	100	100	100	100	100	100	100
5	100	100	100	100	100	100	100	100	100
6	96,3	94,44	77,5	100	88,89	72,5	100	83,33	80
7	100	100	85	100	100	90	100	100	90
8	100	100	90	100	100	90	100	100	90
9	100	100	77,5	100	100	82,5	100	100	75
T(Seg)	67,6	42,9	103	69,1	44,1	105,9	70	44,6	106
Média	99,26	96,11	78,5	98,89	96,67	78,75	99,62	96,11	78,75

Tabela 3.13 : Porcentagens de acerto no reconhecimento em função do número de coeficientes cepstrais. Os resultados para N=14 podem ser vistos nas colunas com tamanho de codebook 256 na Tabela 3.11.

Análise do Número de Parâmetros (N+N+2):
Freq.: 11.025Hz **Superposição**: 66% **Tam. Cbk**: 256
Janela: 20ms **Pré-ênfase**: 0,95 **Núm. Estados**: 5

N.	16			18			20		
Base	Treino	Tes1	Tes2	Treino	Tes1	Tes2	Treino	Tes1	Tes2
0	100	100	92,5	100	94,44	90	100	100	95
1	100	94,44	37,5	100	88,89	45	100	100	50
2	100	100	80	100	100	72,5	100	100	65
3	100	94,44	77,5	100	88,89	70	92,6	72,22	72,5
4	100	100	100	100	100	100	100	100	92,5
5	100	100	97,5	100	100	97,5	100	100	92,5
6	96,3	88,89	75	96,3	100	85	100	100	65
7	100	100	77,5	100	100	87,5	100	100	80
8	100	100	90	100	100	100	100	100	100
9	100	100	87,5	100	100	87,5	100	100	85
T (Seg)	76,7	48,7	116,8	85,9	54,5	130,8	93,4	59,1	141,7
Média	99,62	97,78	81,5	99,62	97,22	83,5	99,26	97,22	79,75

Tabela 3.14 : Porcentagens de acerto no reconhecimento em função do tamanho da superposição. Os resultados para superposição de 66% podem ser vistos nas colunas com N=18 na Tabela 3.13.

Análise do Tamanho da Superposição das Janelas:
Freq.: 11.025Hz **Pré-ênfase**: 0,95 **Núm. Par**.: 18+18+2
Janela: 20ms **Tam. Cbk**: 256 **Núm. Estados**: 5

	0%			33%			50%		
Base	Treino	Tes1	Tes2	Treino	Tes1	Tes2	Treino	Tes1	Tes2
0	100	100	90	100	94,44	90	100	100	92,5
1	100	83,33	60	100	88,89	60	100	100	57,5
2	100	100	85	100	100	65	100	100	55
3	100	88,89	80	100	94,44	62,5	100	77,78	57,5
4	100	100	100	100	100	100	100	100	92,5
5	100	100	100	100	100	100	100	100	97,5
6	96,3	83,33	65	100	88,89	85	100	83,33	77,5
7	100	100	92,5	100	100	87,	100	100	87,5
8	100	100	95	100	100	97,5	100	100	100
9	100	94,44	95	100	100	97,5	100	94,44	85
T(Seg)	30,4	19,4	46,3	46,5	29,3	70,8	59,9	38,1	91,3
Média	99,62	95	86,25	100	96,67	84,5	100	95,56	80,25

Tabela 3.15 : Tabela de confusão para a melhor taxa de acerto encontrada no teste.
(a) para a base teste 1. (b) para a base teste 2.

Melhores resultados:
Freq.: 11.025Hz **Pré-ênfase**: 0,95 **Tam. Cbk**: 128
Janela: 20 ms **Núm. Par**.: 14+14+2 **Núm. Estados**: 4
Superposição: 66%

		Dígito reconhecido (acerto médio de 95%)										
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Média**
Dígito apresentado	**0**	18										100
	1	1	15		1					1		83,33
	2			18								100
	3			1	16			1				88,89
	4					18						100
	5						18					100
	6			1	1		1	15				83,33
	7								18			100
	8									18		100
	9			1							17	94,44

(a)

		Dígito reconhecido (acerto médio de 86,25%)										
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Média**
Dígito apresentado	**0**	36							4			90
	1	1	24				12		1	2		60
	2	1		34	1			1	1	1	1	85
	3	1			32			2	3	2		80
	4					40						100
	5						40					100
	6				3		3	26	7	1		65
	7	2						1	37			92,5
	8					2				38		95
	9					2					38	95

(b)

3.4. Análise da detecção de extremos

Todos os testes anteriores foram feitos com bases de dados sem nenhum erro na detecção dos extremos, ou seja, os dígitos com erro na detecção foram recortados manualmente. Nessa seção aparecem os testes realizados com uma base de dados recortada manualmente (**TrM**, **TesM**) e com uma base de dados com recorte automático (**TrA**, **TesA**). A base com recorte manual foi a mesma que foi utilizada na Seção 3.2. As bases **TrM** e **TrA** são formadas por 20 amostras de cada dígito e as bases **TesM** e **TesA** são formadas por outras 40 amostras de cada dígito.

A Tabela 3.16 mostra o erro no recorte automático. Percebe-se claramente que mais da metade das amostras tiveram um erro no recorte do fim. Existem dois problemas principais que causaram esse erro na detecção. O primeiro é um clique que acontece após o final da palavra. Esse clique é devido ao fechamento dos lábios após o pronunciamento da palavra. O segundo problema é um "sopro" devido ao esvaziamento dos pulmões.

Tabela 3.16 : Mostra o número de dígitos com recorte errado no início e no fim da palavra. O número de sons total usados nesse teste é 200.

	Início	**Fim**				**Bons**
	< 100 ms	**< 100 ms**	**< 300 ms**	**> 300 ms**	**Total**	
0	2	4	4	4	12	8
1	2	1	3	10	14	6
2	0	5	3	2	10	10
3	2	6	5	1	12	8
4	3	4	5	1	10	8
5	1	1	4	1	6	14
6	0	3	4	2	9	11
7	2	1	5	1	7	11
8	2	4	9	1	14	4
9	2	2	7	1	10	10
Total	16	31	49	24	104	90

A Tabela 3.17 mostra a taxa de acerto quando o sistema é treinado com as bases **TrA** e **TrM**. Assim como mostrado na Seção 3.2, quando treinamos com **TrM** e testamos com **TesM** a taxa de acerto é de 98,75%. Porém, quando foi testado com **TesA** a taxa de acerto caiu para 93,5%. Verificou-se que quando foram usadas a base **TrA** no treino e a base **TesA** no teste, o acerto caiu para 96,75%.

Tabela 3.17 : Resultados com as bases com recorte automático(TrA, TesA) e manual(TrM, TesM). Os parâmetros utilizados nesse teste são F=11,025KHz, Pe=0,95, J=20ms, S=66%, N=14+14+2, T. Cbk=128, E=4.

Treino com:	**TrA**				**TrM**			
Teste com:	**TrA**	**TrM**	**TesA**	**TesM**	**TrM**	**TrA**	**TesM**	**TesA**
0	100	100	100	100	100	100	100	87,5
1	100	100	100	100	100	85	100	90
2	100	100	95	97,5	100	100	100	90
3	100	100	97,5	100	100	100	100	87,5
4	100	100	100	100	100	100	100	100
5	100	100	100	100	100	100	100	100
6	100	95	87,5	82,5	100	100	92,5	85
7	100	100	90	87,5	100	100	97,5	100
8	100	95	100	97,5	100	100	100	100
9	100	100	97,5	100	100	100	97,5	95
Média	100	99	96,75	96,5	100	98,5	98,75	93,5

4. Conclusões

Neste trabalho, utilizou-se uma freqüência de amostragem de 11,025 KHz e um fator de pré-ênfase (μ) de 0,95. Os melhores resultados encontrados com as duas bases de dados utilizadas nesse trabalho podem ser vistos na tabela abaixo.

Locutor	Dependente	Independente
Tamanho da janela	18ms a 22 ms	20ms
Superposição	66% e 50%	0%
Número de parâmetros	12 a 14	18
Tamanho do *codebook*	128 e 256	256
Número de estados:	3 e 4	3,4 e 5
Taxa de acerto com a base de treino	100%	99,62%
Taxa de acerto com a base de teste 1	98,75%	95%
Taxa de acerto com a base de teste 2	-	86,25%
Tempo de processamento por dígito	108ms	115ms

Tabela 4.1 : Melhores resultados

Com o sistema criado e com os resultados obtidos, podemos chegar às seguintes conclusões:

- Se o sistema for testado com um microfone diferente do utilizado no treinamento, a taxa de acerto se reduz bastante.

- A captação do ruído de fundo deve ser feita com o microfone colocado nas condições em que ele será utilizado.

- O principal problema da etapa de detecção de início e fim de palavra, e que causa o erro de detecção, é um clique que acontece após o final da palavra. Esse clique é devido ao fechamento dos lábios após o pronunciamento da palavra.

- Outro problema relativo à etapa de detecção é um "sopro" devido ao esvaziamento dos pulmões. Isso é muito comum com palavras terminadas com o fonema /o/. Esse problema reduz a taxa de acerto no reconhecimento.

- Percebeu-se que o tempo gasto no processamento é muito afetado pelo tamanho do codebook e pelo tamanho da superposição. Mas verificou-se, que, em nenhum dos casos, esse tempo impede a realização de um sistema em tempo real.

- A principal causa de erros no reconhecimento neste trabalho é devido ao sistema muitas vezes confundir os dígitos "dois", "três", "seis", "sete" e "um". Uma solução para esse problema foi sugerida na Seção 3.3.

Existe uma infinidade de possibilidades para as diversas variáveis analisadas nesse trabalho. Mas, com a ajuda dos **softwares** criados neste trabalho, os alunos

de reconhecimento de voz poderão realizar trabalhos nessa área com rapidez e eficiência.

Este trabalho pode ser melhorado com a implementação da extração de parâmetros **mel-cepstral**, com o aumento do vocabulário de reconhecimento, com a escolha de modelos para subunidades das palavras e tratamento de fala contínua. Como direcionamento futuro, pode-se trabalhar pelas melhorias desse sistema e na construção de aplicações, utilizando os algoritmos aqui apresentados.

5. Bibliografia

[1] DELLER, J. R. ; PROAKIS, J. G. & HANSEN, J. H. *Discrete-Time Processing of Speech Signals*. MacMillan. 1993.

[2] RABINER, L. R. & JUANG, B. H. *Fundamentals of Speech Recognition*. Prentice Hall. 1993.

[3] KUTWAK, A. B. *Análise da Codificação LPC para Sinais de Fala*. Projeto Final de Curso. UFRJ. 1999.

[4] ESPAIN, C. *Comunicação Pessoal*. Universidade do Porto. Portugal.

[5] RABINER, L. R. *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*. Proceedings of IEEE, vol 77 no. 2, February 1989.

[6] GERSHO, A. & GRAY, R. M. *Vector Quantization and Signal Compression*. Kluwer. 1993.

[7] SOUZA, M. N. *Comunicação Pessoal*. COPPE/UFRJ. Programa de Engenharia Biomédica.

[8] Samsung. *Samsung Voiced*. Telefone com discagem por voz.

[9] IBM. *ViaVoice Pro Millennium*. Editor de textos em português. 1999.
http://www.br.ibm.com

[10] Philips. *FreeSpeech 2000*. Editor de textos em português. 1999.
http://www.speech.philips.com

[11] RAPOSO, E. P. & STEMMER, M. R. *Um Sistema de Reconhecimento de Fala Aplicado a Interação com um Robô*. Anais do XII Congresso Brasileiro de Automática. 1998. pp. 2088-2091.

[12] LIMA, A. A. *Reconhecimento de dígitos Isolados Usando DTW*. Projeto Final de Curso. UFRJ. 1999.

[13] ISHI, C. T. ; PASSOS, R. A. S. & SAOTOME, O. *Análise e Implementação de um Sistema Reconhecedor de Voz*. Anais do XV Simpósio Brasileiro de Telecomunicações. 1997. pp. 6-9.

[14] KLAUTAU, A. ; LEITÃO P. S. ; VIEIRA, A. & TAKITA, K. *Reconhecimento de palavras isoladas para aplicação em automação industrial*. Anais do X Congresso Brasileiro de Automática. 1994. pp. 1252-7.

[15] MINAMI, M. ; ALENS, N. & SANCHES, I. *Sistema Reconhecedor de Palavras Isoladas, com HMM-VQ, Múltiplos Livros de Códigos e Quantização Vetorial de Energia*

para a Linha Telefônica. Anais do XV Simpósio Brasileiro de Telecomunicações. 1997. pp. 332-5.

[16] SOLEWICZ, J. A. & CALOBA, L. P. *Um Sistema Integrável para o Reconhecimento de Palavras Isoladas em Português*. Anais do IX Congresso Brasileiro de Automática. 1992. pp. 851-5.

[17] SANBUICHI, C. A. & AGUIAR NETO, B. G. *Reconhecimento de Palavras Isoladas Independente de Locutor para Língua Portuguesa do Brasil*. Anais do IX Simpósio Brasileiro de Telecomunicações. 1991. pp. 7.1.1-5.

[18] SILVA, F. J. F. *Implementação em Tempo Real de um Reconhecedor de Dígitos Isolados Independente do Locutor*. Anais do IX Simpósio Brasileiro de Telecomunicações. 1991. pp. 7.2.1-5.

Apêndice 1:
Descrição dos programas

Neste apêndice, mostra-se uma descrição do funcionamento dos programas criados e utilizados ao longo deste trabalho. Em todos estes programas não foram desenvolvidas rotinas rigorosas de tratamento de erros. Portanto, as faixas de valores previstas para as opções devem ser respeitadas.

O sistema é uma aplicação desenvolvida para uma plataforma *Windows 9x*. Ele reconhece os dígitos de zero até nove. A interface com o usuário foi feita em *Delphi 3.* As funções que processam o sinal de voz e fazem o reconhecimento foram desenvolvidas no *Borland C++ 5* e transformadas em *DLLs*.

A.1. Sistema de reconhecimento em tempo real

O funcionamento do programa é simples. A janela principal é apresentada na Figura A.1. Primeiro, aperta-se o botão **Inicializar Sistema**. Após a detecção de ruído de fundo pelo sistema, pode-se apertar o botão **Reconhecer novo dígito** e falar uma palavra ao microfone. Será tocado o som detectado e ele mostrará o dígito mais provável. Durante todo o processo, a situação do sistema aparece no campo **Status**, ou seja, esse campo mostra uma mensagem, informando o que o sistema espera do usuário.

Para a realização de testes do sistema, ele mostra o módulo do logaritmo das probabilidades encontradas em cada modelo HMM. Assim, é possível obter uma avaliação melhor nos casos em que o sistema errar o dígito.

Também é possível ver a palavra que foi detectada, apertando-se o botão chamado **Ver sinal**. Com isso, aparece uma janela que mostra o sinal que foi recortado. Podendo até mesmo ser ouvido.

Pode-se ainda configurar os parâmetros necessários para a extração de parâmetros. Para isso, aperta-se o botão **Opções**, fazendo aparecer uma janela como mostra a Figura A.2. Nesta janela, encontram-se todos os parâmetros que podem ser modificados. Qualquer mudança afeta as funções que estão dentro de uma *DLL*. Mas para que as opções surtam efeito, deve ser gerado um codebook e devem ser treinados os modelos HMM dos dígitos, levando-se em consideração os valores utilizados nessas constantes. O codebook e os modelos são gravados nos arquivos *centros.dat* e *modelos.dat*, respectivamente.

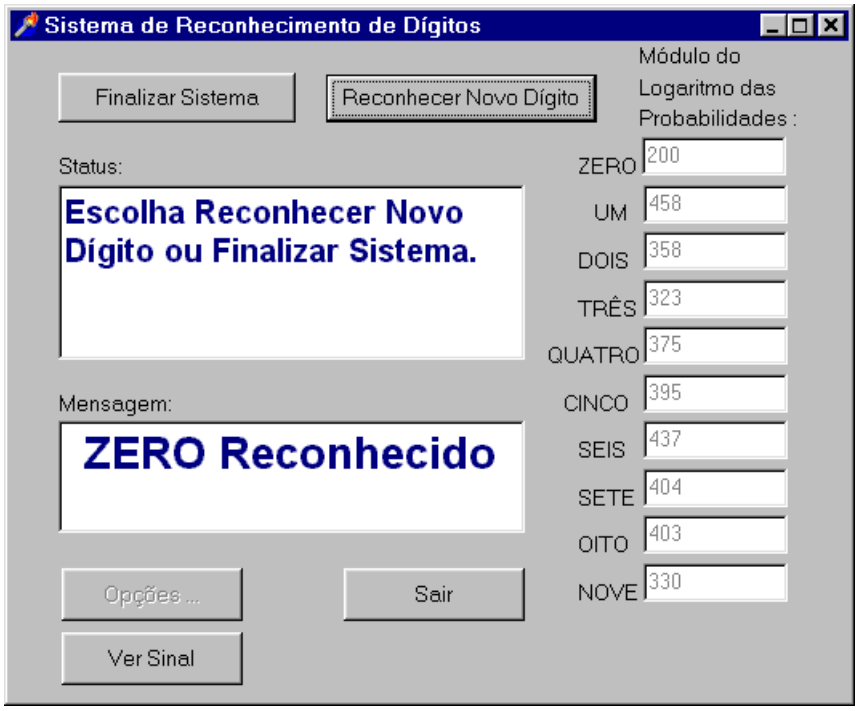


Figura A.1 : Janela do programa principal.

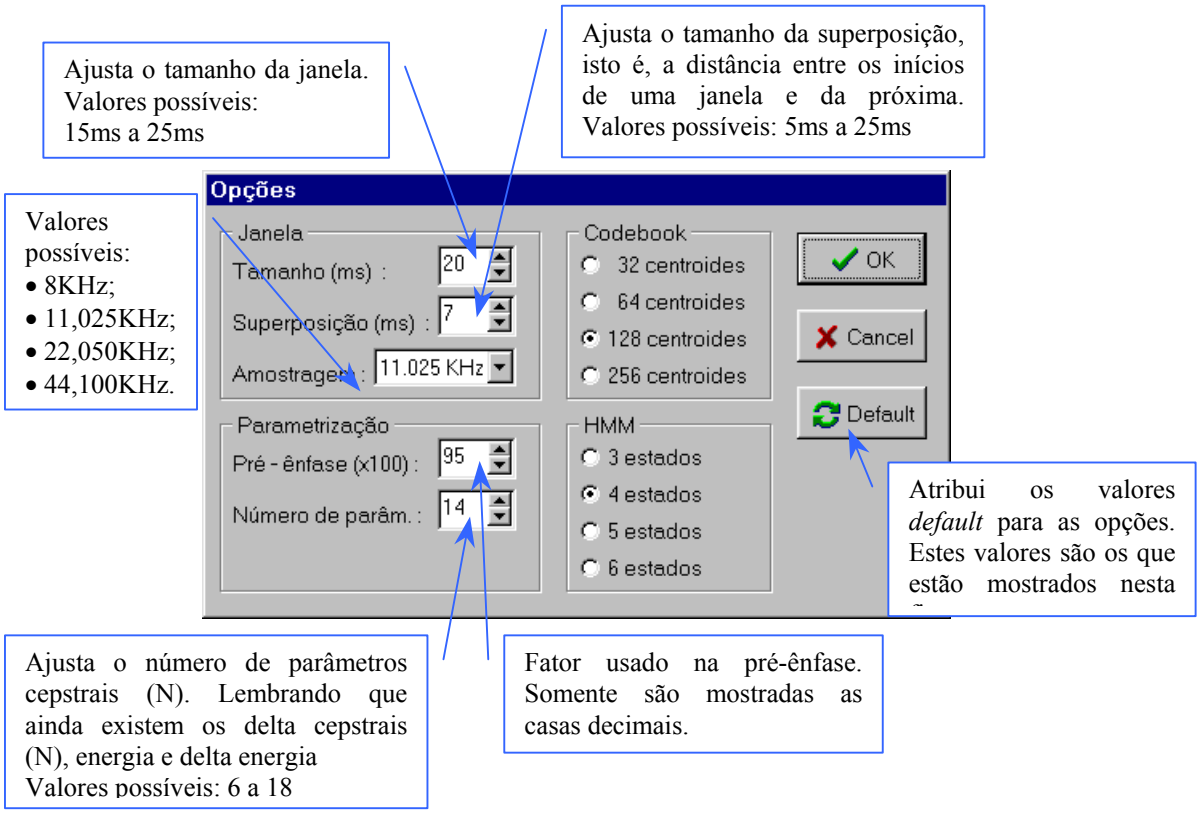


Figura A.2 : Janela de opções.

Todo o processamento de voz foi feito em linguagem C no *Borland C++ 5.0* e compilados em uma *DLL* (*hmm.dll*) do *Windows 9x*. A utilização é feita através de duas funções: *configuracao* e *reconhece*. A primeira inicializa todas as constantes do programa. Estas configurações são ajustadas na janela de opções do programa (Figura A.2).

A segunda função deve ser chamada sempre que se desejar reconhecer uma palavra. Ela recebe um ponteiro para o vetor de dados que contém a palavra e o seu tamanho. Dentro dela são chamadas as funções: *extraiparametros*, *quantiza*, e *viterbi*; as quais também estão presentes na *DLL*.

Abaixo mostra-se o cabeçalho das funções em C e como elas devem ser declaradas dentro do *Delphi*. Com essas declarações, pode-se utilizar o código da *DLL* como se fossem funções comuns, mas tomando-se o cuidado de colocar a *DLL* no mesmo diretório que o programa em *Delphi*.

Declaração em C:

- ```
  extern "C" __declspec(dllexport)
  void WINAPI configuracao(tipo pe, inteiro tj, inteiro ol, inteiro np,
                inteiro nc, inteiro nh, inteiro nm) {
  código ...
  }
  ```
- ```
  extern "C" __declspec(dllexport)
  inteiro WINAPI reconhece(inteiro *s,inteiro tam) {
   código ...
  }
  ```

Declaração em *Delphi*:

- ```
  procedure configuracao(pe: double; tj, ol, np,nc,nh,nm: smallint);
    stdcall;external 'hmm.dll' name 'configuracao';
  ```
- ```
  function startrec(s: psinal;tam: smallint): smallint;
    stdcall;external 'hmm.dll' name 'reconhece';
  ```

O sistema de detecção de extremos de palavra é feito em *Delphi 3*. A aquisição de dados é realizada através de funções do *Windows 9x*. Desse modo, não é necessário nenhum conhecimento prévio sobre o tipo de placa de som que o microcomputador está utilizando, deixando o controle de mais baixo nível para o próprio *Windows*.

O arquivo que contêm o código da detecção chama-se *detpalavra.pas* e a sua utilização é feita através dos procedimentos mostrados abaixo:

- *procedure InicializaSistema* : chamado pelo botão *Inicializa Sistema* da janela principal. Inicializa as variáveis, capta ruído de fundo e abre canal de voz.
- *procedure DetectaPalavra(var image : image_wave_file)* : começa a detecção de uma palavra e para apenas quando uma palavra é detectada. Grava a palavra detectada na variável *image*.
- *procedure FinalisaSistema* : fecha canal de voz.
- *procedure TocaPalavraDetectada(image : image_wave_file)* : Toca arquivo *wave* que estiver na memória e for passado como parâmetro da função.

A.2. Extração de parâmetros:

Este programa, cuja interface está apresentada na Figura A.3, realiza a extração de parâmetros e cria o conjunto de treinamento que será utilizado para gerar o codebook. Para isso, ajusta-se as opções (Figura A.4). Não se deve esquecer de ajustar os mesmos valores durante o teste do sistema. Ao apertar o botão **iniciar**, o programa começa a calcular todos os parâmetros para cada janela do sinal. Ao terminar o processamento, os parâmetros extraídos devem ser salvos em um arquivo texto.

- Exemplo de arquivo de entrada com 2 amostras de cada dígito:

.\Waves\zero1.wav
.\Waves\zero2.wav
.\Waves\um1.wav
.\Waves\um2.wav
.\Waves\dois1.wav
.\Waves\dois2.wav
.\Waves\três1.wav
.\Waves\três2.wav
.\Waves\quatro1.wav
.\Waves\quatro2.wav
.\Waves\cinco1.wav
.\Waves\cinco2.wav
.\Waves\seis1.wav
.\Waves\seis2.wav
.\Waves\sete1.wav
.\Waves\sete2.wav
.\Waves\oito1.wav
.\Waves\oito2.wav
.\Waves\nove1.wav
.\Waves\nove2.wav

- Formato do arquivo de saída:

{Tempo gasto: Hora Minuto Segundo Milisegundo}
{número de janelas na primeira wave}
    {lista de parâmetros (cada linha corresponde a uma janela)}
{número de janelas na segunda wave}
    {lista de parâmetros (cada linha corresponde a uma janela)}
...
{número de janelas na última wave}
    {lista de parâmetros (cada linha corresponde a uma janela)}

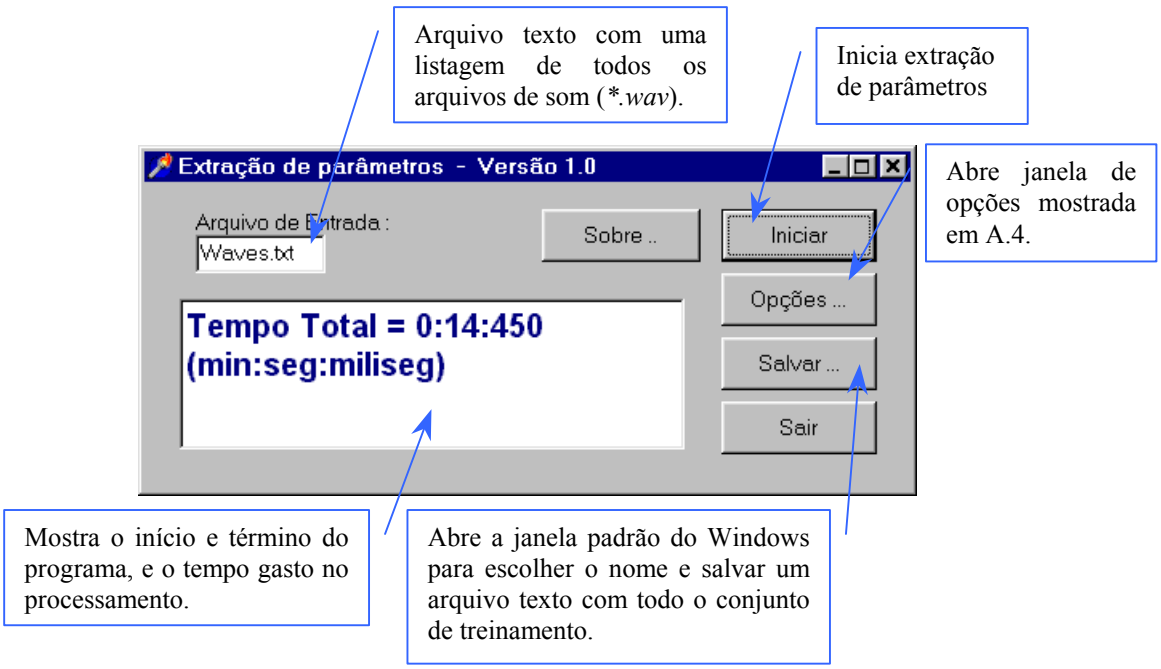


Figura A.3 : Janela do programa de extração de parâmetros.

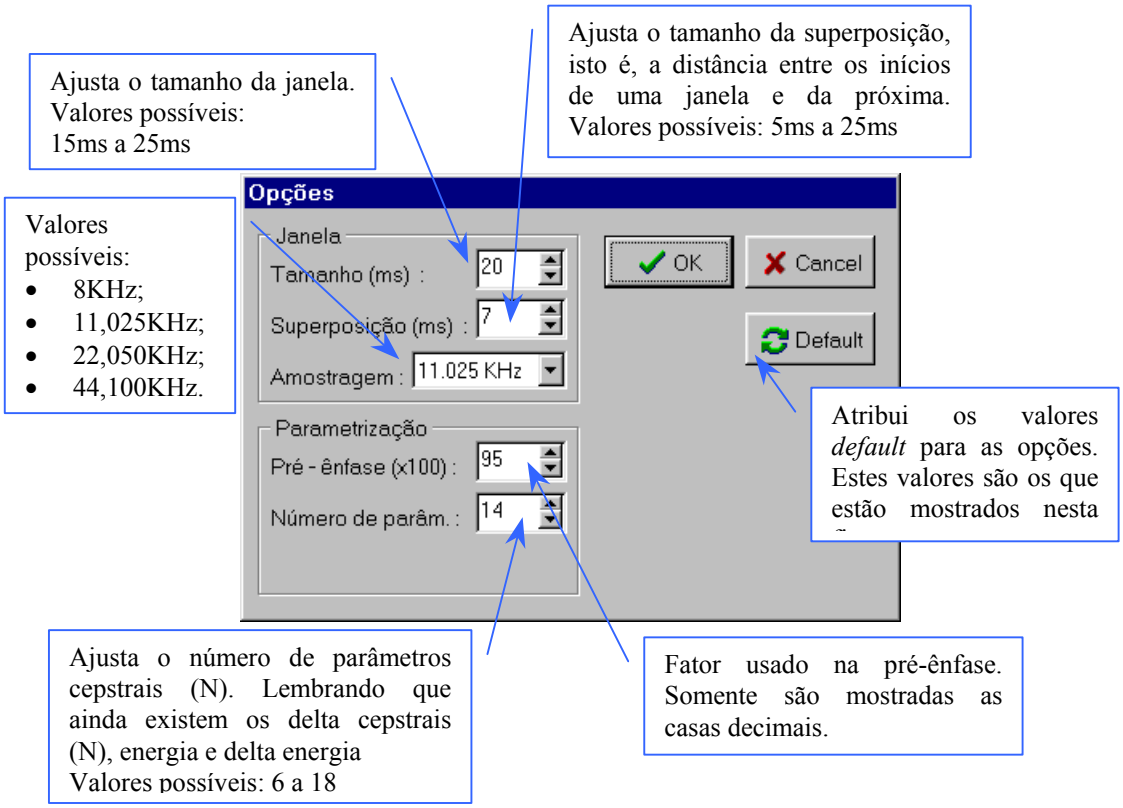


Figura A.4 : Janela de opções do programa.

A.3. Geração do Codebook

Este programa gera um codebook para cada tamanho entre o inicial e o final (2^1=2 , 2^2=4 ,2^3=8 , ... , 2^{10}=1024). Isso poupa tempo quando se quer codebooks de vários tamanhos com a mesma base de dados. Ele utiliza o conjunto de treinamento criado pelo programa anterior como entrada. A interface deste programa aparece na Figura A.5. Após ajustar as opções, aperta-se o botão **iniciar** e aguarda-se o término do processamento. No final, teremos arquivos do tipo *cbkXX.txt* e *centrosXX.dat*, onde *XX* é o tamanho do codebook (2,4,8,...,1024). Os arquivos *cbkXX.txt* são utilizados pelo próximo programa (Seção A.4) para criar os modelos. As informações contidas nele é mostrada abaixo. Os *centrosXX.dat* são arquivos binários contendo o codebook gerado para o tamanho *XX* correspondente. Ele está no formato adequado para os programas que testam (Seções A.1 e A.5).

- Formato dos arquivos *cbkXX.txt* :

{Tempo gasto: Hora Minuto Segundo Milisegundo}
{número total de janelas} {número de centróides} {número de parâmetros} {epsi} {limiar de término} {distância média total}
{o mesmo codebook que aparece no arquivo centrosXX.dat correspondente}
{vetores quantizados} {distância ao respectivo centróide} (cada linha corresponde a uma janela)

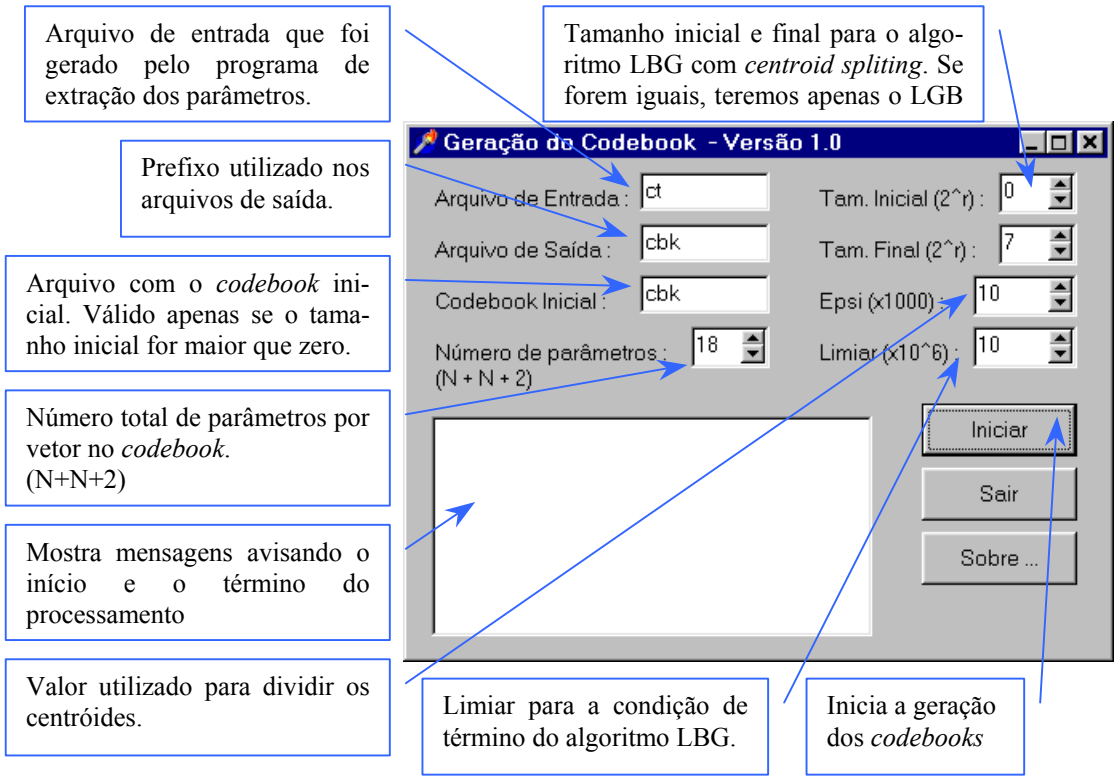


Figura A.5 : Janela do programa que gera os *codebooks*.

A.4. Treinamento dos modelos HMM

Este programa cria o arquivo *modelos.dat* com os modelos HMM de cada dígito. Para isso, ele utiliza como entrada o conjunto de treinamento quantizado pelo programa anterior (*cbkXX.txt*). Primeiro, as opções devem ser ajustadas. Então, aperta-se o botão **iniciar** e aguarda-se que surja a mensagem de término do processamento. O treinamento é realizado com o algoritmo de Baum-Welch.

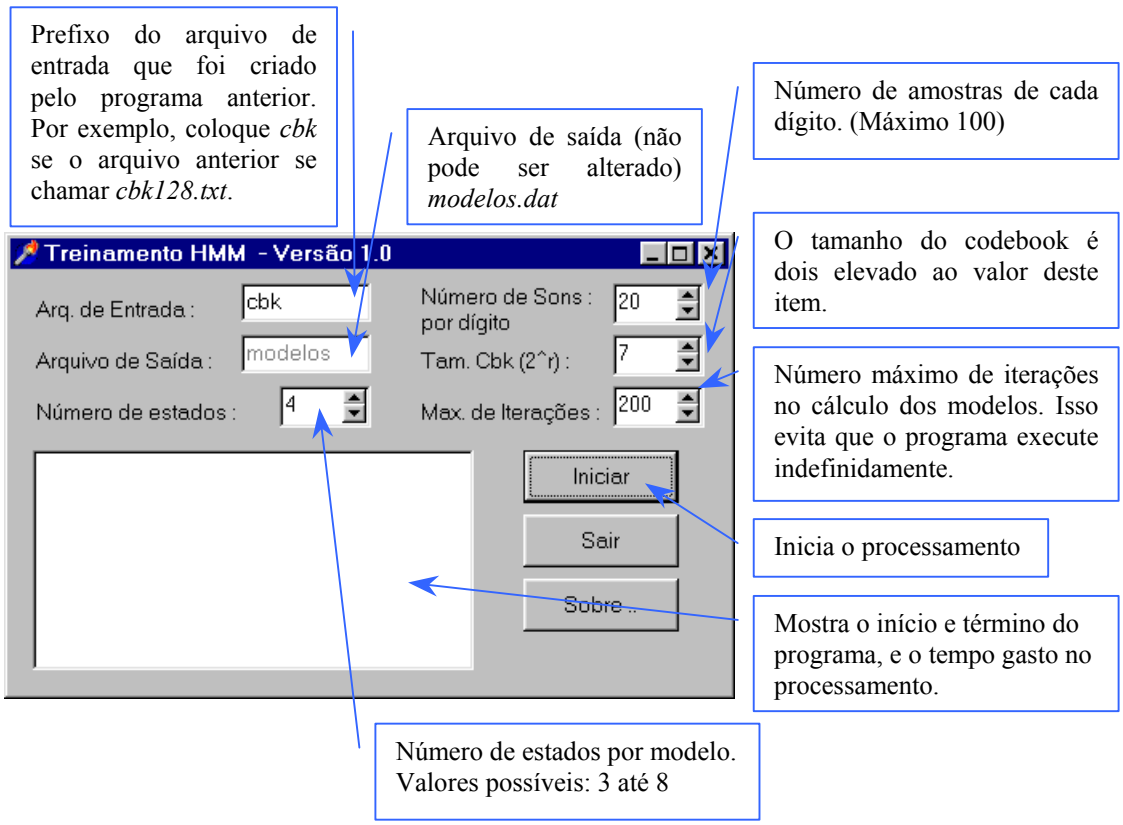


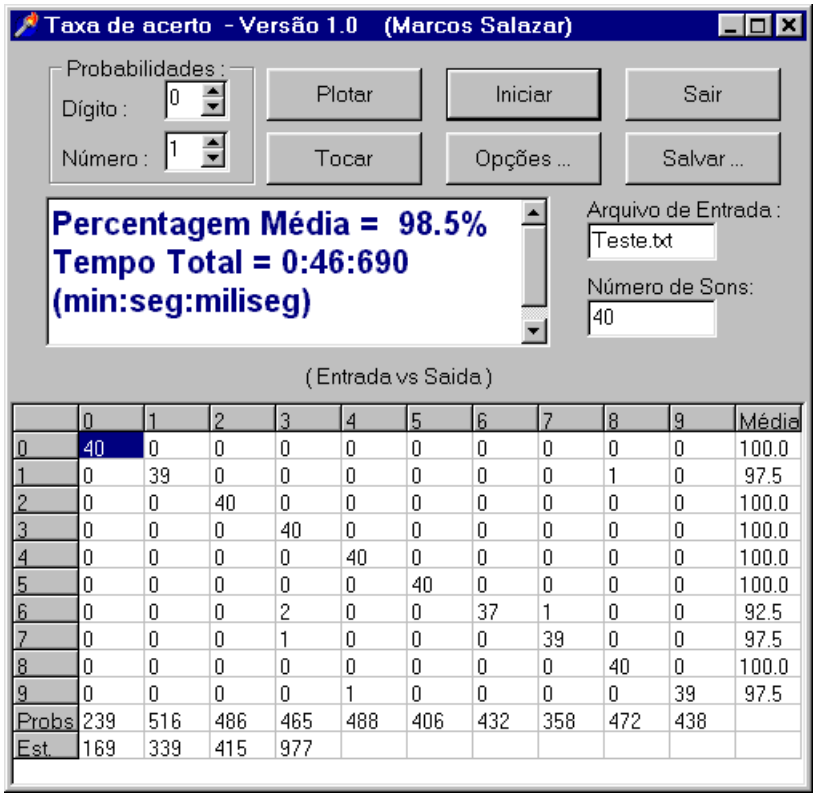
Figura A.6 : Janela do programa que realiza o treinamento dos modelos HMM.

A.5. Teste da taxa de acerto

Este programa permite realizar o teste do sistema para uma lista de arquivos de som. Calculando a taxa de acerto para cada dígito. Ele recebe um arquivo de entrada do mesmo tipo usado pelo programa que gera o conjunto de treinamento (seção A.2). Inicialmente, ajusta-se as opções (Figura A.8), o arquivo de entrada e o número de sons por dígito. Então, aperta-se o botão **iniciar** e aguarda-se. Ao término, diversas informações serão mostradas. Será mostrado o tempo gasto, uma tabela de confusão (dígito apresentado vs dígito reconhecido), a média de acerto por dígito e a média de acerto total do sistema.

Além disso, pode-se verificar as probabilidades encontradas por dígito, ver a forma de onda e tocá-lo. Para isso, escolhe-se o dígito e o número da amostra do dígito (canto superior esquerdo da janela).

Também é possível ver na última linha da tabela, a soma de todas as ocorrências de um determinado estado para todas as amostras de um dígito (o dígito também deve ser ajustado no canto superior esquerdo).



	0	1	2	3	4	5	6	7	8	9	Média
0	40	0	0	0	0	0	0	0	0	0	100.0
1	0	39	0	0	0	0	0	0	1	0	97.5
2	0	0	40	0	0	0	0	0	0	0	100.0
3	0	0	0	40	0	0	0	0	0	0	100.0
4	0	0	0	0	40	0	0	0	0	0	100.0
5	0	0	0	0	0	40	0	0	0	0	100.0
6	0	0	0	2	0	0	37	1	0	0	92.5
7	0	0	0	1	0	0	0	39	0	0	97.5
8	0	0	0	0	0	0	0	0	40	0	100.0
9	0	0	0	0	1	0	0	0	0	39	97.5
Probs	239	516	486	465	488	406	432	358	472	438	
Est.	169	339	415	977							

Figura A.7 : Janela principal do programa que realiza as estatísticas.

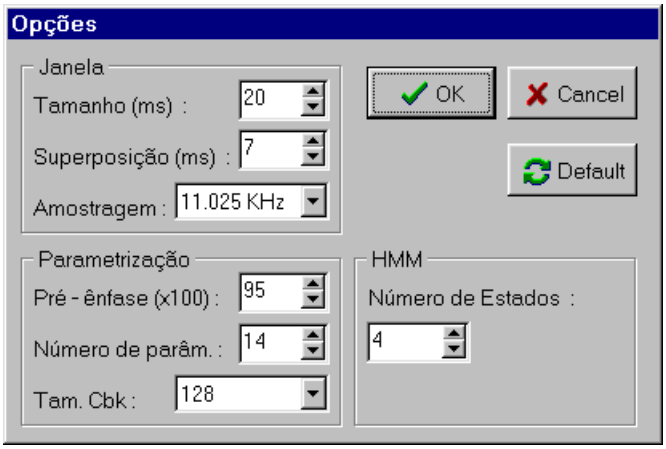


Figura A.8 : Janela de opções.