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

Projeto Final

Reconhecimento de dígitos isolados usando DTW

Orientadores:

Prof. Fernando Gil Resende Vianna Junior
Prof. Sérgio Lima Netto

Aluno:

Amaro Azevedo de Lima

PREFÁCIO

O objetivo deste trabalho é implementar um sistema de reconhecimento de sinais de fala cujas aplicações no mundo atual são infinitas. Podemos citar exemplos como: automatização de sistemas para pessoas deficientes, editores de texto que transcrevem tudo que foi falado, criar uma interface mais amigável entre homem/computador e discagem automática através de comandos de voz. Para não pensarmos que isto só ocorrerá num futuro muito distante, este último exemplo já existe em telefones celulares comercializados aqui mesmo no Brasil.

Já existem sistemas de reconhecimento bastante eficientes porém nenhum deles é totalmente independente de limitações e nem funciona perfeitamente com 100% de acerto. Podemos concluir então que esta área ainda tem muito potencial para se desenvolver e é de grande importância científica, pois, um sistema de grande eficiência e segurança poderia gerar um grande impacto na relação homem-máquina, talvez não mais utilizaríamos teclas somente a voz.

Mais especificamente, o trabalho consiste em implementar um sistema para reconhecimento de dígitos isolados usando a técnica de DTW ("*dynamic time warping"*), que é um método determinístico utilizado no reconhecimento de sinais, na verdade, este método é baseado na comparação do sinal com padrões previamente estabelecidos. Funciona bem para aplicações não muito complexas e de pequeno vocabulário.

Este trabalho tem por intuito ser utilizado como uma referência didática para futuros alunos que trabalhem no grupo de Processamento de sinais de fala da Universidade Federal do Rio de Janeiro. A seguir, abordaremos os tópicos de cada capítulo.

No capítulo 1, daremos uma introdução aos conceitos referentes ao processamento de sinais de fala, conceitos relativos especificamente ao reconhecimento e um diagrama de blocos do algoritmo implementado.

No capítulo 2, iremos propor uma solução para o recorte de sinais de fala também chamado de detecção de extremos("*endpoint*"). A idéia é recortar o sinal de fala de modo a capturarmos somente a parte do sinal que contém informação relativa à fala, eliminando toda e qualquer forma de ruído que apareça antes e depois do sinal de fala. Explicaremos alguns algoritmos que têm por objetivo solucionar este problema.

No capítulo 3, explicaremos técnicas para representar os sinais de fala. Esta representação é necessária para diminuirmos o tamanho da seqüência que representa o sinal e facilitar a comparação com outras seqüências. O objetivo é representar uma seqüência numérica (sinal de fala discreto no tempo) por uma seqüência numérica bem menor, sem que haja perda expressiva de informação de fala.

No capítulo 4, explicaremos com detalhes o algoritmo DTW. Este algoritmo basicamente quantifica o resultado da comparação entre sinais de fala de tamanhos diferentes e mostraremos o funcionamento dos blocos de treinamento e reconhecimento do sistema implementado neste trabalho, pois, estes utilizam o algoritmo DTW como unidade básica.

No capítulo 5, apresentaremos e comentaremos os resultados de vários experimentos (testes) realizados com o sistema.

No capítulo 6, faremos algumas conclusões baseadas nos resultados do capítulo 5 sistema e mencionaremos ainda possíveis rumos de trabalhos futuros que queiram dar continuidade ao assunto.

No apêndice, explicaremos como usar as funções implementadas no Matlab para reconhecer sinais de fala.

1 - Introdução

A interface entre a língua falada e o computador é um tópico que tem fascinado os engenheiros e os cientistas por mais de 50 anos. Para muitos, a habilidade de se falar livremente com a máquina representa o desafio maior para a compreensão da produção e da percepção da fala.

A entrada de sinal de fala em um computador engloba muitas tecnologias e aplicações diferentes como as descritas na figura abaixo.

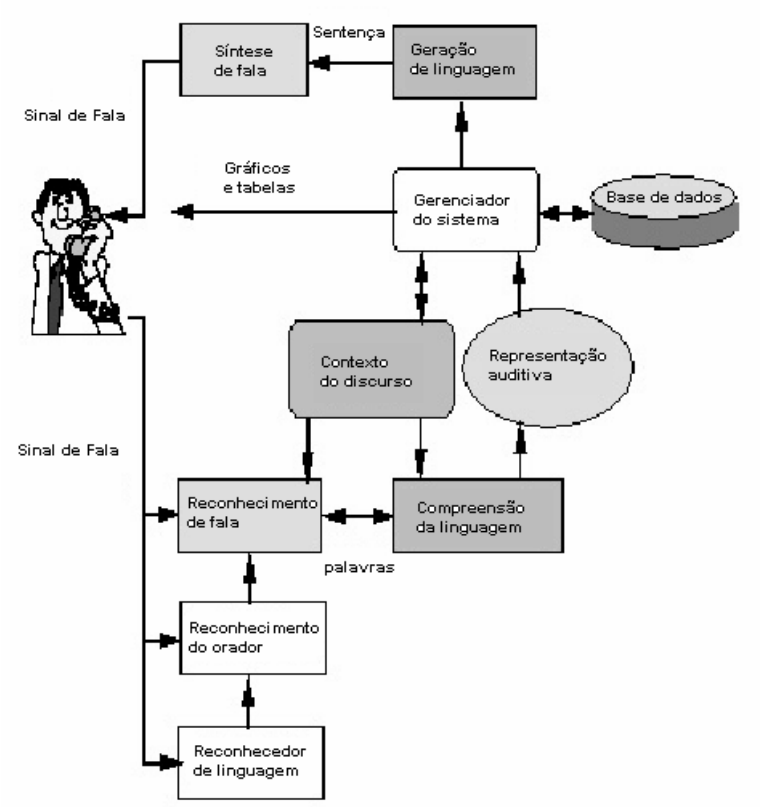


Figura 1: Tecnologias para interfacear a linguagem falada.

Em alguns casos, como é mostrado na parte inferior da figura, não se esta interessado na compreensão/entendimento do conteúdo lingüístico, e sim, identificar o interlocutor ou a linguagem por ele utilizada. O reconhecimento do interlocutor pode envolver a identificação de um interlocutor específico dentro de um conjunto de pessoas conhecidas, ou a verificação da identidade de um usuário, deste modo habilitando o acesso controlado a certos locais (e.g., uma sala de computadores) e serviços (e.g., acesso bancário via telefone).

Quando nós pensamos em "falar" com um computador, a primeira imagem que vem em nossas mentes é o problema de reconhecimento da fala, a conversão de sinais acústicos em um fluxo de palavras. Depois de muitos anos de pesquisas, o reconhecimento da fala está começando a ultrapassar o limiar do mundo virtual para o prático.

O reconhecimento da fala envolve vários componentes tecnológicos. O primeiro é o sinal digitalizado que deve ser transformado em um conjunto de medidas, representação matemática do sinal. Também existem técnicas que possibilitam os sistemas alcançarem a robustez perante o uso de transdutores e variações ambientais e se adaptarem a estas variações. O próximo passo é modelar apropriadamente os vários sons falados. A técnica mais usada para esta modelagem é chamada de HMM ("*hidden Markov models"*). A procura pela resposta final envolve o uso das restrições da língua. O reconhecimento de sinal de voz é um problema cada vez mais desafiador; o que nos leva sempre a pesquisar e desenvolver métodos mais apurados com o objetivo final de fazer um sistema que seja o mais parecido possível com o sistema natural de linguagem. É claro que a acurácia do sistema dependerá da aplicação por ele requisitada. Porém, quanto mais próxima a aplicação for da linguagem humana, menos restrições ou conhecimento prévio do sistema precisaremos ter para utilizá-lo.

1.1 - Definindo o problema

Reconhecimento de sinais de fala é o processo de converter um sinal acústico, capturado através de um microfone ou telefone, em um conjunto de palavras. As palavras reconhecidas podem ser utilizadas como comandos, dados de entrada como num sistema de consulta de saldo bancário por telefone ou também como entrada para um sistema mais complexo de processamento lingüístico que objetiva o entendimento da fala pela máquina.

Os sistemas de reconhecimento podem ser caracterizados por vários parâmetros:

Tabela 1: Parâmetros típicos usados para caracterizar a complexidade dos sistemas de reconhecimento de fala.

Parâmetros	Faixa de aplicação
Modo de fala	Palavras isoladas x palavras contínuas
Estilo de fala	Fala por leitura x fala espontânea
"Envolvimento" (do orador)	Dependente do orador x independente do orador
Vocabulário	Pequeno (<20 palavras) x grande (>20.000 palavras)
Modelo de linguagem	Estado finito x sensível ao contexto
Perplexidade	Pequeno (<10) x grande (>100)
SNR(Relação sinal/ruído)	Alto (>30dB) x baixo (<10dB)
Transdutor	Microfone x telefone

Um sistema de reconhecimento de palavras isoladas requer que o interlocutor faça uma pequena pausa entre o pronunciamento das palavras, ao passo que o reconhecimento de palavras conectadas não requer esta pausa.

Geração espontânea de palavras numa frase, ou numa conversa contém variações na pronúncia e na intonação da palavras, É diferente pronunciar

isoladamente por exemplo, a palavra "azul", e observar como fica a pronúncia dela na frase, "De quem é esta casa azul ?". Neste último caso parece que a palavra "casa" e a palavra "azul" se fundem formando uma nova palavra "casazul", o que mostra evidentemente uma maior dificuldade em se reconhecer palavras geradas espontaneamente do que palavras geradas, por exemplo, através da leitura de um texto.

Alguns sistemas requerem envolvimento (participação) do interlocutor, o que significa que o interlocutor deve fornecer alguns exemplos de sua fala antes de usar o sistema, e os sistemas que não requerem este tipo de treinamento são ditos independente do interlocutor.

Alguns outros parâmetros dependem da especificidade da tarefa a ser realizada. O reconhecimento se torna mais difícil quando o vocabulário é grande ou possui muitos sons parecidos. Quando o caso é de uma seqüência de palavras, as limitações da língua em questão e as normas gramaticais são utilizadas para restringir a gama de combinações de palavras. Os modelos mais simples podem ser especificados como uma rede de estados finitos onde as possibilidades são dadas de forma explícita, e os modelos mais gerais, que se aproximam da linguagem natural, são especificados em termo de uma gramática sensível ao contexto (as limitações são dependentes do contexto).

Uma medida popular de dificuldade da tarefa, combinando o tamanho do vocabulário e o modelo da língua é a perplexidade, é definida como a média geométrica do número de palavras que pode vir depois de outra palavra após ser aplicada a modelagem da língua. Existem também parâmetros externos que podem afetar o sistema de reconhecimento da fala, como o ruído ambiente e o tipo e posicionamento do microfone.

O reconhecimento de sinal de voz é um problema extremamente difícil devido às extensas possibilidades de variação associadas ao sinal. Primeiro, a percepção acústica dos fonemas, a menor unidade de som pela qual as palavras são compostas, é altamente dependente do contexto no qual ele aparece. As possibilidades de variações fonéticas podem ser exemplificadas pelo fonema /s/ em "suor", e em "casa" já com som /z/, em português do Brasil. No que diz respeito às fronteiras das palavras, as variações contextuais podem ser um grande problema como por exemplo, "sossobra" e "só sobra". Segundo, as possibilidades de variações acústicas podem ser resultado de mudanças no meio, como a posição e as características do transdutor (exemplo, microfone). Terceiro, possibilidades de variações adquiríveis pelo interlocutor podem ser resultado de mudanças do estado emocional e físico do interlocutor, da taxa de fala (quão veloz se fala) ou da qualidade da voz. Finalmente, diferenças de formação socio-lingüística, dialeto e na forma e tamanho do aparelho articulador da fala podem contribuir para as possibilidades de variações inerentes ao interlocutor.

1.2 - Componentes típicos de um sistema de reconhecimento de fala

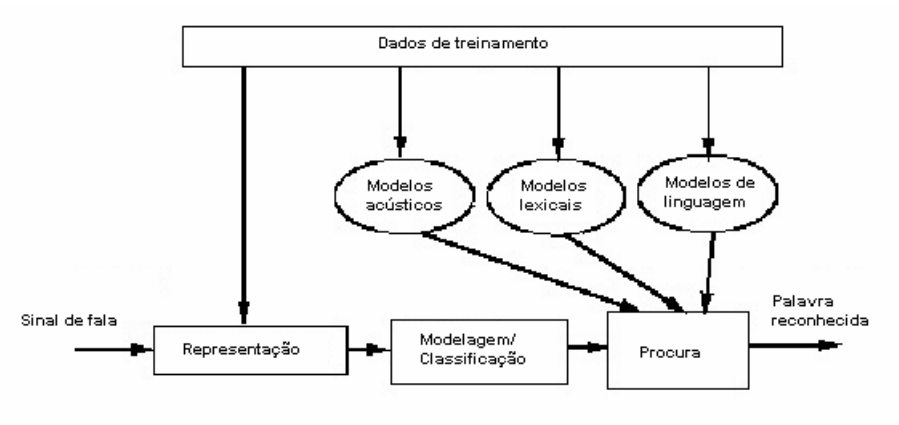


Figura 2: Componentes de um sistema de reconhecimento de fala típico.

Podemos observar na figura acima os mais importantes componentes de um sistema de reconhecimento típico. O sinal digitalizado é primeiramente transformado em um conjunto de medidas ou características úteis em uma certa taxa fixada, geralmente um segmento a cada 10-20 ms. Estas medidas são então usadas para procurar a palavra candidata mais provável fazendo uso de restrições impostas pelos modelos acústicos, lexical e de linguagem. Por intermédio deste processo, os dados de treinamento são usados para determinar os valores do modelo de parâmetros.

Sistemas de reconhecimento de fala tentam modelar as fontes de variabilidade descritas acima de várias formas. No que diz respeito a representação de sinal, os pesquisadores têm desenvolvido representações que ressaltam claramente importantes caraterísticas independentes do orador e não as características dependentes do orador do sinal. Na área de fonética acústica, a variabilidade do interlocutor é modelada geralmente através de técnicas estatísticas aplicadas a um grande número de dados. Os efeitos do contexto lingüístico na área da fonética acústica são normalmente tratados por treinamento de modelos de fonemas separadamente em diferentes contextos, isto é chamado de modelagem dependente da acústica.

A variabilidade da palavra pode ser tratada pela pronúncia alternada da palavra em representação conhecida como rede de pronúncia. A pronúncia alternada de palavras habitual, assim como os efeitos do dialeto e do sotaque, são tratados por algorítmos de procura para encontrar caminhos alternados de fonemas através destas redes. Modelos estatísticos da língua, baseados na estimativa da ocorrência da freqüência na seqüência de palavras, costumam ser usados para conduzir a procura através da mais provável seqüência de palavras.

O mais predominante paradigma de reconhecimento, pelo menos nestes últimos 15 anos é conhecido como HMM. O HMM é um modelo duplamente estocástico, no qual a geração de cadeias de fonemas e de segmentos são

representadas probabilisticamente como processos de Markov. Redes Neurais também têm sido usada para estimar a pontuação dos frames, estas pontuações são integradas em sistemas que utilizam arquitetura HMM, sendo conhecido como sistema híbrido.

1.3 - Representação do sinal de fala

No reconhecimento automático de sinal de voz baseado em estatística, a forma de onda do sinal é amostrada numa taxa que varia de 6.6kHz a 20kHz e processada para produzir uma nova representação como uma seqüência de vetores contendo valores que chamamos de parâmetros. Estes vetores são compostos tipicamente por 10-20 parâmetros e são geralmente computados a cada 10-20 ms (maiores detalhes sobre este assunto serão abordados no capítulo 3 deste trabalho). Estes parâmetros são usados em sucessivos estágios na estimação da probabilidade daquela parte da forma de onda analisada correspondendo a um evento fonético particular tido como referência. Na prática, a representação e a estimação da probabilidade interagem fortemente.

As representações objetivam preservar a informação necessária para determinar a identificação fonética, sendo ao mesmo tempo indiferentes a fatores como diferença no interlocutor, efeitos introduzidos pelo meio (canal) e fatores paraligüísticos, como o estado emocional do interlocutor. As representações atuais são sensíveis a vibração das cordas vocais (som vozeado/não vozeado), mas tentam ignorar os efeitos devido as variações na freqüência de vibração das mesmas (Fo) e quase sempre são derivadas dos termos em espectro de potência e ignoram as estruturas de fase, pois, os nossos ouvidos são de um modo geral, insensíveis aos efeitos da fase.

O espectro de freqüência é quase sempre representado em escala logarítmica, ou seja, quando o ganho aplicado ao sinal varia, a forma do espectro de potência é preservada, sendo somente deslocado para cima ou para baixo. E as mais complicadas filtragens lineares causadas, por exemplo, por salas acústicas ou variações nas linhas telefônicas, que aparecem como convolução em forma de onda, como multiplicação em espectro de potência linear, se tornam simplesmente uma constante aditiva no espectro de potência logarítmica. Existem problemas com a representação das partes de baixa energia, pois, o logaritmo de 0(zero) é menos infinito. Logo, é necessário limitar uma faixa numérica para prevenir excessiva sensibilidade de baixa energia na partes do espectro dominada pelo ruído.

Geralmente usamos um filtro pré-ênfase simples, para dar um aumento de 6 dB/oitava para fazer com que a média do espectro de sinal fique quase achatada.

O espectro é derivado de superposição de partes de formas de onda pré-enfatizadas (em geral, 25 ms), e passando por uma função janela de forma

de bell e aplicando transformada de Fourier. Aparecem então harmônicos indesejáveis múltiplos de Fo. Isto pode ser reduzido agrupando os conjuntos vizinhos formando então, aproximadamente 20 bandas de freqüência, onde estas bandas são limitadas por fronteiras de mais de 1 kHz de acordo, com a técnica de escala de freqüência de mel, refletindo assim a freqüência de resolução do ouvido humano.

Desde que a forma do espectro seja suave, os níveis de energia em bandas adjacentes tenderão a ser correlacionados, porém ao se remover a correlação mantendo a informação útil, pode-se reduzir o numero de parâmetros. Pode-se usar a transformada dos cossenos para fazer a conversão do conjunto de energia logarítmica em um conjunto de coeficientes cepstrais (que são altamente não correlacionados), deste modo um conjunto de sinal correlacionado pode se tornar um conjunto não correlacionado.

Para muitos sinais de voz em condições acústicas favoráveis, uma boa aproximação seria modelar o trato vocal por um filtro só de pólos. Uma técnica conhecida como LPC (*"linear predictive coding"*) ajusta os parâmetros do filtro só de pólos para o espectro de voz. Isto gera um método alternativo popular de derivar os coeficientes cepstrais. Existe também a técnica PLP (*"perceptual linear prediction"*), que é um método que combina LPC com um banco de filtros. Existem muitos outros métodos e técnicas não mencionadas aqui, pois, esta parte só serve para nos dar uma idéia superficial sobre o assunto, devido ao fato do tópico principal do trabalho ser o reconhecimento de sinal de voz.

1.4 - Algumas técnicas de reconhecimento

As técnicas mais usadas em reconhecimento são DTW, HMM e NN (Redes Neurais). Abaixo daremos uma rápida explicação sobre estas duas primeiras técnicas mencionadas.

DTW é um método usado no reconhecimento de sinal de voz. Existem dois métodos nos quais quase todos os nossos atuais algoritmos de reconhecimento são baseados. O primeiro método a ser visto é método no qual usamos padrões que serão comparados com o sinal obtido de modo a se gerar uma resposta se "casam" ou não. Melhor dizendo, se o reconhecimento foi positivo ou não. O grande problema deste método, no qual se enquadra o DTW, é alinhar o sinal falado, isto é, alinhar suas características, com as dos padrões já existentes. Para esta grande tarefa computacional, usamos o princípio da programação dinâmica, que como se pode observar, acabou originando o nome: esticando ou comprimindo (em inglês, *"warping*"), porque como já fora mencionado, o sinal é esticado ou comprimido no tempo de modo a ser alinhado com o molde (padrão) e dinâmico (em inglês, *"dynamic"*), porque usamos a programação dinâmica como instrumento, para chegar ao resultado, daí observamos a origem do nome DTW. O segundo método é baseado numa aproximação estocástica e HMM é o mais usado. Este pode ser considerado

uma generalização do DTW e logicamente utiliza de forma mais pesada a programação dinâmica.

1.5 - Conceitos básicos de sinais de fala

Um sinal de fala é formado através de uma excitação no trato vocal e de forma geral é composto por dois tipos de sons – os vozeados e não vozeados. O som vozeado é resultado da vibração das cordas vocais quando excitadas por um fluxo de ar pulsátil. Estes são os sons das vogais e de alguma consoantes como por exemplo: B, D, M, N, L e R. Já os sons não vozeados são produzidos por um fraco e curto fluxo de ar na parte anterior da cavidade oral (lábios) e mantendo as cordas vocais em descanso. Estes representam os sons como F (“faca”), S (“sapo”) e X (“axé”).

Mostraremos abaixo alguns exemplos de formas de onda de sons vozeados e não vozeados. A palavra falada foi “faca”, porém só estão representadas na figura as partes referentes a sílaba “fa” e separadamente “f” e “a”.

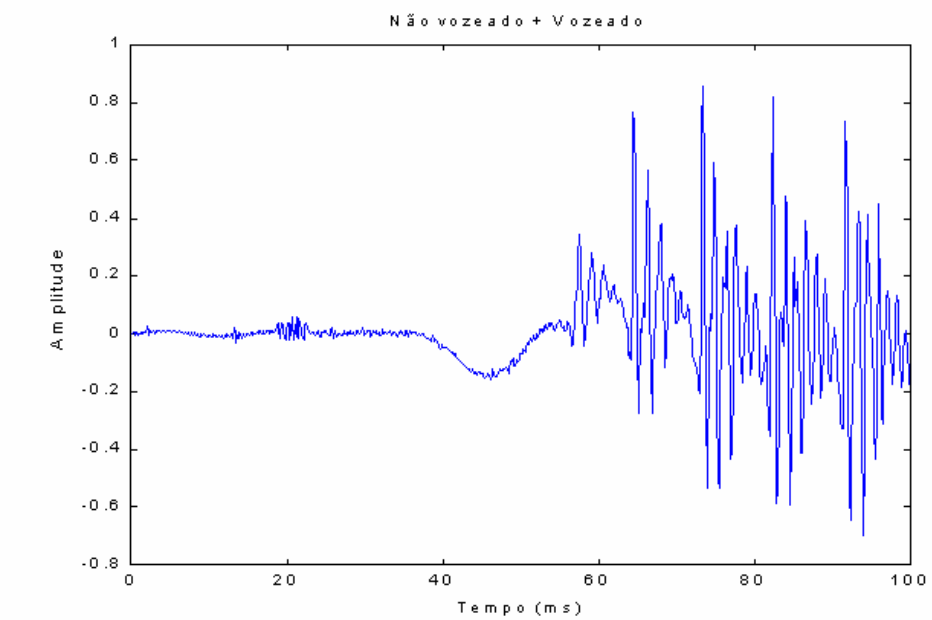


Figura 3: Segmento referente a “fa” da palavra “faca”.

Observamos a presença de uma parte não vozeada e uma outra parte vozeada no segmento de fala da fig. 3. Já os segmentos representados nas figuras 4 e 5 adiante, são respectivamente a parte não vozeada do segmento acima (fig. 3), no caso o “f” e a parte vozeada que é o “a”.

Na fig. 4 notamos um grande número de componentes de alta freqüência, porém sem nenhuma periodicidade de tal modo que se assemelham mais com a forma de ruído. Também vale a pena ressaltar que a energia em sinais vozeados são muito mais altas que em sinais não vozeados, basta observarmos as amplitudes dos sinais das figuras 4 e 5.

Analisando a fig. 5 notamos a lenta variação em baixa freqüência do sinal vozeado e podemos perceber que este sinal é aproximadamente periódico. A mais baixa freqüência de oscilação nesta representação é chamada de *freqüência fundamental* ou *freqüência de pitch*.

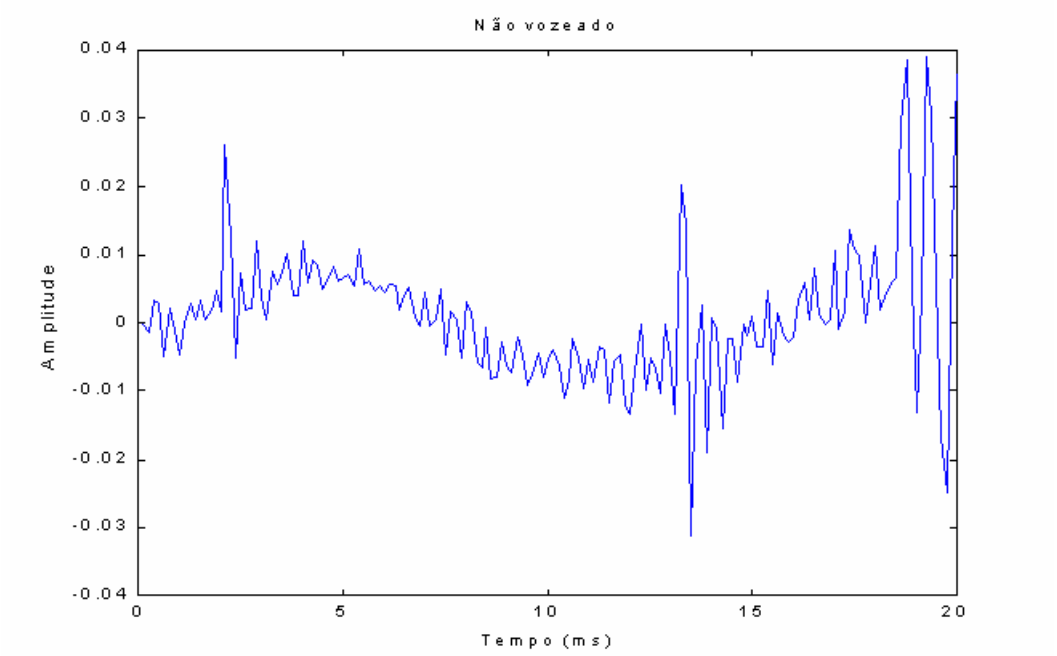


Figura 4: Segmento referente a “f” da palavra “fa”.

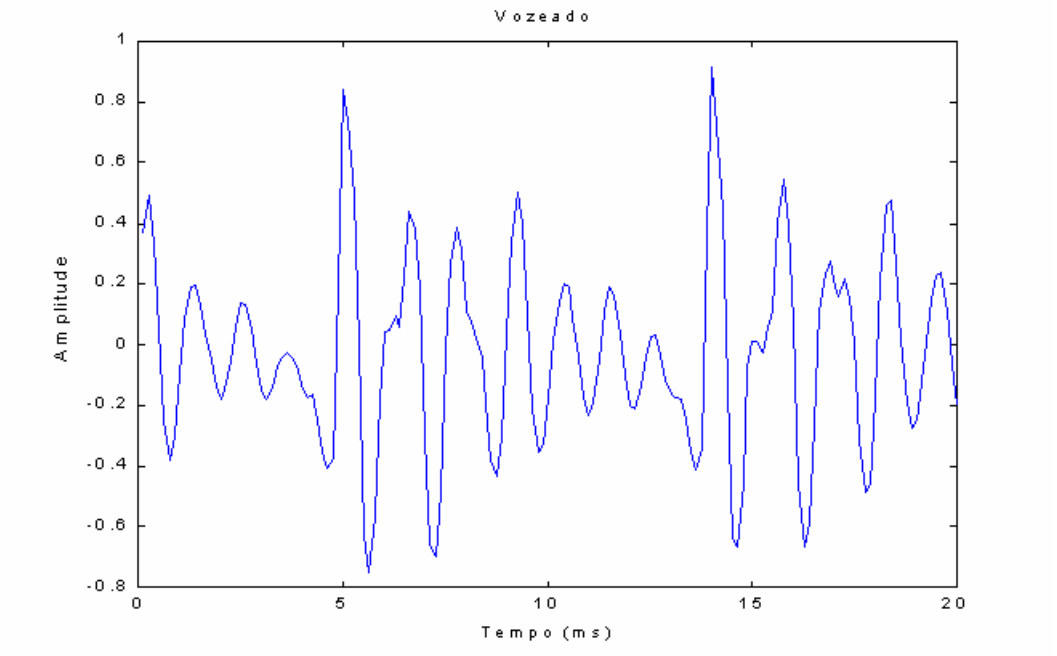


Figura 5: Segmento referente a “a” da palavra “fa”.

Podemos observar com clareza nas figuras 4 e 5 que os sinais quando analisados dentro de janelas de tempo bem pequenas na faixa de 20 ms, eles se mantém bem “comportados”, ou melhor dizendo, eles podem ser considerados como quase estacionários. Esta característica de estacionariedade é devido a lenta variação da forma do trato vocal e da estacionariedade da excitação que passa por ele para produzir o som. No caso de sinais não vozeados seria algo parecido com ruído branco e para vozeados seria algo parecido com uma senóide cuja freqüência seja igual a freqüência de pitch.

1.6 - Amostragem de sinais analógicos

O teorema da amostragem para sinais que possuam energia finita e banda limitada, diz que:

1 - Um sinal de energia finita e banda limitada, que não tenha componentes em freqüência maior que W hertz, é inteiramente descrito pela especificação dos valores do sinal em instantes de tempo separados por $\frac{1}{2W}$ segundos.
2 - Um sinal de energia finita e banda limitada, que não tenha componentes em freqüência maior que W hertz, pode ser completamente recuperado através do conhecimento de suas amostras tomadas a uma taxa de 2W por segundo.

A taxa de amostragem de 2W é chamada de taxa de Nyquist , e $\frac{1}{2W}$ é chamado de intervalo de Nyquist. O teorema descrito acima nos diz que ao amostrarmos sinais analógicos com uma taxa igual ou superior a taxa de Nyquist, teremos ainda acesso a todas as informações contidas no sinal original, assim podemos representar um sinal analógico por seqüências numéricas, ou seja, sinal discreto.

No caso da freqüência de amostragem ser menor que a freqüência de Nyquist, gerará “aliasing”, ou melhor, superposição no espectro do sinal amostrado, de modo que na reconstrução do sinal original, quando este passar por um filtro PB(passa-baixas) nos proporcionará um sinal recuperado com distorções devido a superposição ocorrida na amostragem do sinal analógico.

A figura a seguir mostra um diagrama de blocos com os componentes básicos para a implementação de um conversor de sinal analógico para discreto. O filtro PB é para limitarmos a banda do sinal entre –W e W hertz, e com isto podermos ter em mãos todas as informações contidas nesta faixa do sinal analógico original. O circuito amostrador nada mais é que uma multiplicação na faixa de interesse do sinal analógico por um trem de impulsos espaçados por $\frac{1}{2W}$ segundos. Na verdade este esquema abaixo converte um

sinal contínuo tanto em amplitude como no tempo, em um sinal contínuo em amplitude e discreto no tempo.

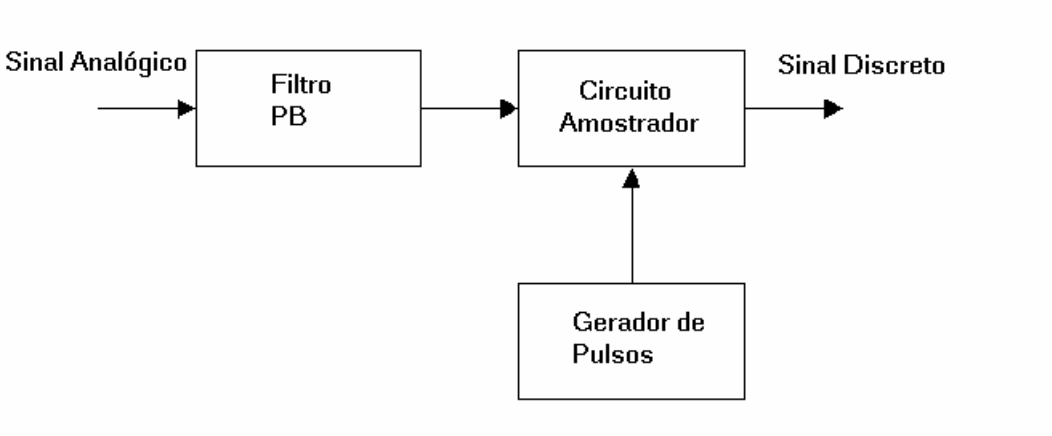


Figura 6: Diagrama de blocos de um conversor analógico-discreto.

No caso de sinais de fala usados em telefonia a faixa de interesse está entre 300 Hz e 3400 Hz, na qual utiliza-se taxas de amostragem de 8 kHz. Já para sinais de áudio é necessário representarmos freqüências de até 20 kHz, onde utiliza-se uma taxa de amostragem de 44.1 kHz (taxa usada em CD). No caso dos sinais utilizados neste trabalho, foi utilizada uma freqüência de amostragem de 8 kHz, e usamos programas já existentes para a gravação destes sons, pois, o Matlab (ferramenta principal utilizada na implementação deste trabalho) não nos possibilita implementar funções que atuem diretamente na placa de som. Os programas usados foram o Sound'LE e o WaveStudio que fazem parte do kit do Creative Multimídia.

1.7 - Janelamento

Os sinais de fala precisam ser segmentados em frames para que estes possam ser analisados como sinais estacionários e assim possibilitando o processamento em tempo real. Os sinais são segmentados em frames da ordem de 20 ms, o que significa 160 amostras por frame quando a freqüência de amostragem (Fs) usada é 8 kHz.

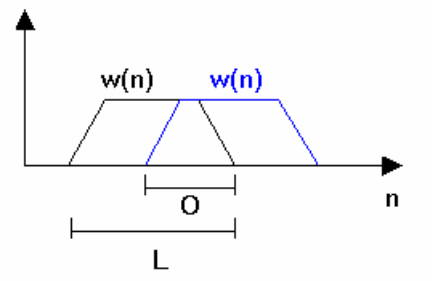


Figura 7: **L** é o comprimento da janela e **O** comprimento de superposição.

A figura 7 acima mostra como se deve fazer uma segmentação de sinal de fala, e como os parâmetros de segmentação que influenciam no processamento final. Estes parâmetros são: comprimento do sinal de fala **L** e comprimento de superposição **O** (*"overlap"*). Na verdade consideramos sempre o valor percentual de superposição, no caso, $p = (O/L) \times 100\%$. Exemplificando: dizer que usamos um overlap de 30%, significa que os 30% finais de amostras de um frame estão sendo consideradas como os 30% iniciais de amostras do frame subseqüente. A superposição de janelas pode melhorar muito o processamento final, pois, os sinais de fala são contínuos e quando janelamos e processamos os segmentos separadamente, desconsideramos a continuidade de um frame para outro, podendo esta ser fundamental no processamento a ser executado.

Outro parâmetro de grande importância é o tipo de janela que usaremos, na segmentação do sinal. A princípio, a idéia seria usar uma janela retangular que funciona muito bem quando o nosso interesse é preservar as características temporais do sinal, porém este tipo de janela promove um corte abrupto no sinal, o que no espectro gerará componentes de alta freqüência. No espectro de freqüência o sinal será distorcido, perdendo sua forma original em espectro.

O problema da escolha do tipo de janela, é sempre uma relação custo benefício de acordo com a aplicação que estamos usando, ou seja, é sempre uma escolha do tipo de janela que funciona melhor para a aplicação de meu interesse, pois, sempre haverá distorções ou no tempo, ou na freqüência, ou em ambos.

Portanto, as janelas precisam pelo menos atender a dois objetivos básicos, para que o espectro de sinal não fique muito distorcido. Estes são:

1. Possuir uma estreita faixa de passagem no lóbulo central.
2. Grande atenuação no lóbulos laterais.

O primeiro é porque uma faixa estreita no lóbulo central, já será suficiente para captar a resolução de detalhes mais abruptos do espectro do sinal de fala e o segundo é para diminuir a influência de ruído devido a "aliasing" no sinal a ser analisado. Existem vários outros tipos de janelas além da retangular, que são bastante utilizadas, como por exemplo, as janelas de Hamming, Hanning e Blackman. Todas estas com cortes mais suaves no domínio do tempo e com maiores atenuações dos lóbulos laterais que a retangular.

Na página seguinte mostraremos as formas de ondas das diferentes janelas no domínio do tempo e da freqüência. Observando que apesar das janelas tipo Hanning, Hamming e Blackman distorcerem bastante o sinal no tempo, elas proporcionam respostas muito melhores no domínio da freqüência quando comparadas com a janela retangular, que não distorce o sinal no tempo. A janela que mais se popularizou foi a janela de Hamming, porém vale a pena ressaltar novamente que a escolha do tipo de janela nada mais é que uma relação custo/benefício entre as resoluções espectral e temporal.

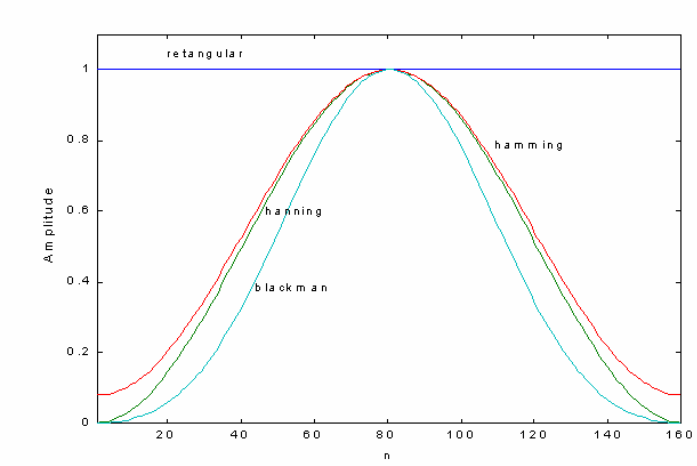


Figura 8: Tipos de janelas utilizadas no domínio do tempo.

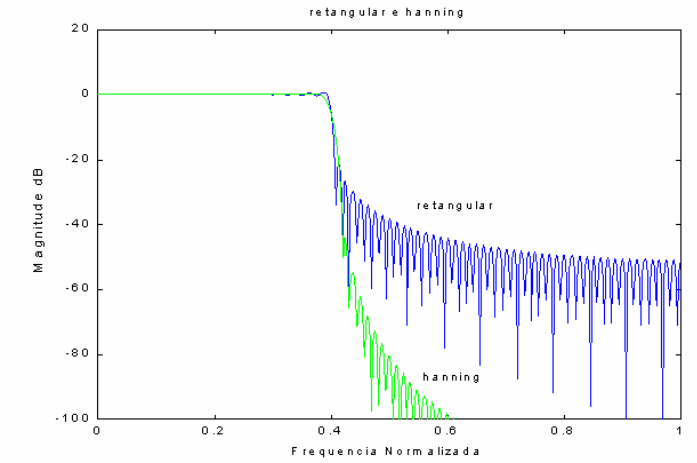


Figura 9: Espectro de freqüência das janelas retangular e hanning.

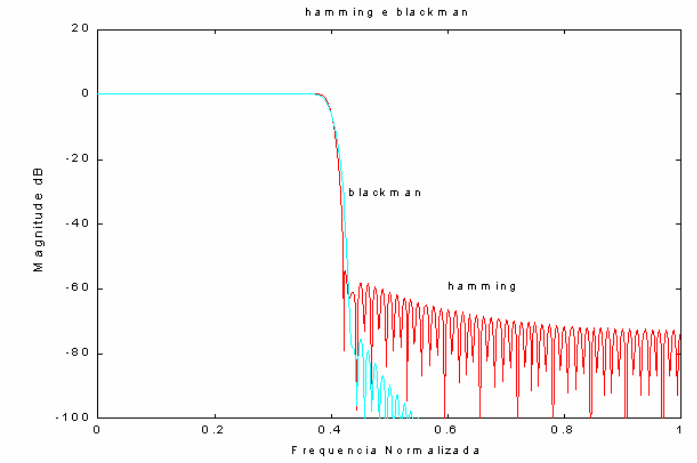


Figura 10: Espectro de freqüência das janelas hamming e blackman.

1.8 - Definições de medidas em sinais segmentados

Como já mencionado, os sinais precisam ser segmentados para que possamos assumir sua estacionariedade e com isto aplicarmos certas análises que nos possibilitam processá-lo mais rápida e eficientemente.

Primeiro mostraremos a definição de frame usada neste trabalho, que é a mesma utilizada no livro de Deller [1].

Frame:

$$f_S(n;m) \overset{\text{def}}{=} s(n) \cdot w(m-n)$$

Na definição acima s(n) é o sinal a ser segmentado, w(n) é a janela usada para segmentar o sinal e esta tem comprimento N. O índice m denota o instante de tempo final de cada janela. O índice s de f_S geralmente é suprimido, pois, só designa o sinal que estamos segmentando.

Agora redefiniremos alguns outros conceitos para sinais segmentados, claro que tomando por base suas definições originais.

DTFT:

Definição para sinais não segmentados:

$$\widetilde{S}(w) = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} s(n) e^{-jwn}$$

Redefinindo para sinais segmentados:

$$S_S \overset{\text{def}}{=} \sum_{n=m-N+1}^{m} s(n) w(m-n) e^{-jwn}$$

DFT:

Definição para sinais não segmentados:

$$S(k) = \sum_{n=0}^{N-1} s(n) e^{-2\pi kn/N}, \; k = 0, ..., N-1$$

$$\text{com } \; 0 \le n \le N-1.$$

Redefinindo para sinais segementados:

$$S(k;m) = \begin{cases} \sum_{n=m-N+1}^{m} f(n;m) e^{-jk(2\pi/N)n}, & k = 0, ..., N-1 \\ 0, & \text{outros valores} \end{cases}$$

1.9 - Diagrama de blocos do sistema de reconhecimento implementado

Este diagrama de blocos que será apresentado abaixo é bastante parecido com a figura 2, porém este tem relação direta com a implementação realizada neste trabalho. Podemos então representar o sistema de reconhecimento implementado por vários blocos que executam processamentos sucessivos. Estes são:

1. Bloco de detecção de extremos.
2. Bloco de extração de parâmetros.
3. Bloco de treinamento.
4. Bloco de reconhecimento.

O bloco de detecção de extremos nada mais é que a realização de recortes no sinal de fala de modo a manter somente as informações de fala propriamente dita eliminando o ruído de fundo (ruído ambiente).

O bloco de extração de parâmetros é uma forma de representar, as amostras do sinal com um número menor de coeficientes. Por exemplo: no caso de usarmos uma freqüência de amostragem de 8 kHz e tamanho de segmento de 20 ms, teremos 160 amostras por segmento, o que geraria um grande trabalho computacional para realizar o processamento. Já representando este segmento por um número menor de coeficientes, poderemos agilizar e facilitar o processamento e a transmissão quando esta for necessária.

O bloco de treinamento representa o processamento realizado num conjunto de sinais de treinamento com o intuito de encontrar um centróide para cada classe. Os sinais de treinamento são grupos de sinais de cada classe (palavra a ser reconhecida) já devidamente recortados. O objetivo é encontrar o sinal que melhor representa a classe, para que este possa ser usado na comparação feita pelo bloco de reconhecimento.

O bloco de reconhecimento é onde geramos o reconhecimento do sinal através da comparação com o centróide de cada classe, e o que gerar melhor resultado será reconhecido como a classe em questão.

Vale a pena ressaltar que os blocos de treinamento e reconhecimento utilizam internamente um bloco que pode ser chamado de algoritmo de reconhecimento, que é bastante parecido com o bloco de reconhecimento porém, com pequenas modificações. Este algoritmo serve para comparar sinais segmento a segmento e no final nos dá o resultado do quão os sinais são parecidos. Esta comparação é realizada através dos parâmetros extraídos dos sinais.

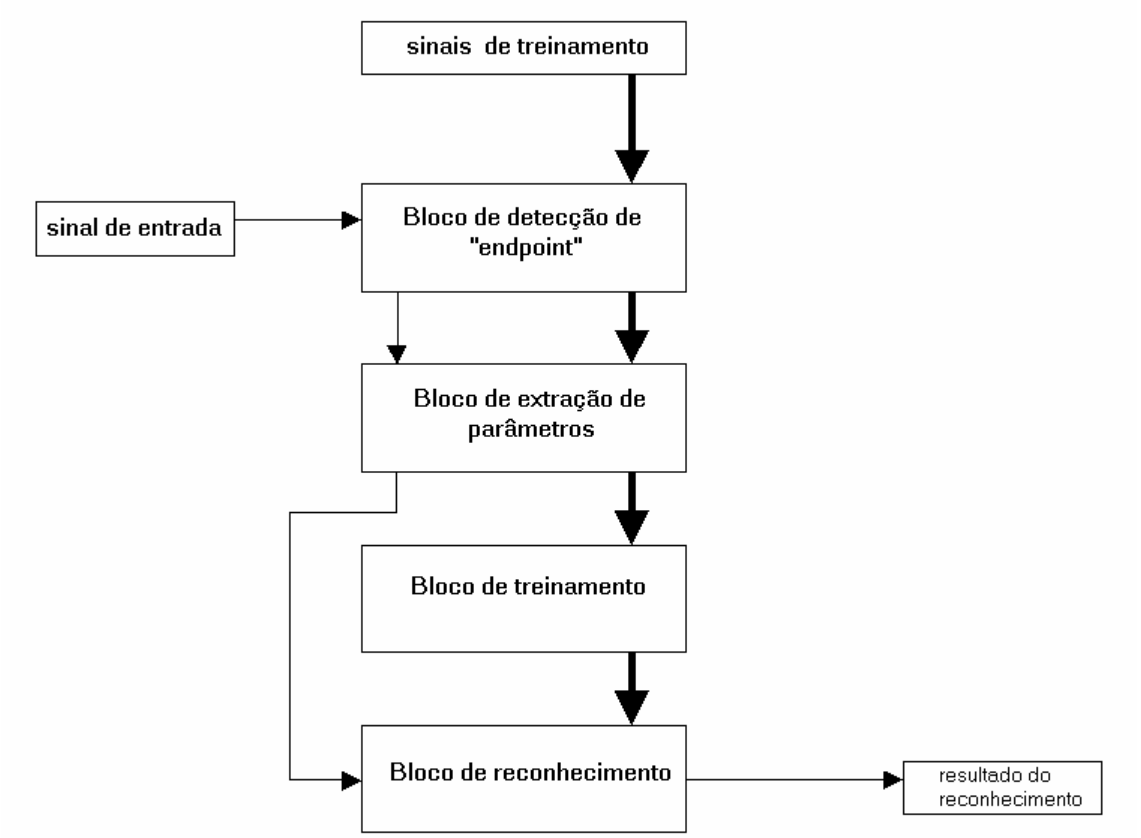


Figura 11: Diagrama de blocos do sistema de reconhecimento implementado.

2. Detecção de extremos (*"endpoint"*)

Antes de entrarmos propriamente no mérito do recorte de sinais de fala, é necessário introduzirmos alguns conceitos e definições primordiais para que a compreensão do algoritmo de detecção de endpoint seja clara. Estes conceitos estão descritos abaixo.

2.1 - Medidas de cruzamentos por zero ("zero-crossing")

É o número de vezes que uma seqüência muda de sinal, ou seja, quantas vezes a forma de onda do sinal cruza o eixo onde a amplitude do sinal é igual a zero. Esta medida é definida por:

$$Z_s(m) = \frac{1}{N} \cdot \sum_{n=m-N+1}^{m} \frac{\left|\text{sgn}\{s(n)\} - \text{sgn}\{s(n)\}\right|}{2} \cdot w(m-n),$$

onde

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

2.2 - Medidas de Energia e Potência

Para finalidades práticas, estas duas medidas carregam as mesmas informações. Definiremos abaixo respectivamente as medidas de energia e de potência:

$$P_s(m) = \frac{1}{N} \cdot \sum_{n=m-N+1}^{m} s^2(n)$$

e

$$E_s(m) = \sum_{n=m-N+1}^{m} s^2(n)$$

Para fins computacionais, a medida mais simples é $E_s(m)$, pois, tem uma divisão a menos, deste modo sendo mais simples e mais rápida computacionalmente que a medida de potência $P_s(m)$.

2.3 - Detecção de extremos

Detecção de extremos nada mais é que detectar o início do sinal de fala e o final do mesmo. A idéia por trás desta detecção é recortar o sinal de fala de forma a se considerar somente as informações relativas a palavra falada propriamente dita, ou seja, toda a informação necessária para a compreensão

da fala, retirando o ruído que aparece tanto antes quanto depois da palavra pronunciada.

Esta detecção é particularmente importante, pois, numa próxima etapa iremos codificar o sinal de fala de modo a representar uma seqüência de por exemplo, 160 pontos (amostras por segmento) por uma seqüência bem menor de por exemplo 39 coeficientes (pontos), onde poderemos usar vários tipos de codificações. Porém, os mais usados e que mais se destacam em ordem de importância são os coeficientes LPC, cepstrais e mel-cepstrais. Devido a isto, não seria coerente representarmos segmentos onde só teríamos sinal de ruído através de coeficientes que foram “projetados” para representar de forma bastante eficiente os sinais de fala. Além de gerar futuras comparações entre segmentos que não corresponderiam propriamente a comparações entre sinais de fala. Estaríamos comparando ruído com sinais de fala, o que poderia nos dar resultados indesejáveis.

O problema de detecção de extremos poderia ser bastante simples se não houvesse trechos não-vozeados nas palavras, pois, estes costumam ser os maiores responsáveis por erros no “recorte” da palavra. As principais causas destes problemas são proporcionadas por fricativas fracas ou fricativas vozeadas no final das palavras por exemplo, o /f/ e o /s/ na palavra “facas”, plosivas fracas (/t/, /b/), sons nasais (“mãe”), e semivogais (“pai”), no final das palavras.

2.3.1 - Algoritmo #1

Este algoritmo foi implementado por mim, e segue a mesma idéia que será apresentada no algoritmo #2, porém com certas diferenças, já que este algoritmo fora implementado antes que eu tomasse conhecimento do algoritmo proposto por Rabiner e Sabur, supra mencionado. Primeiro faz-se uma procura por possíveis extremos através da energia. Começamos pelo segmento 1 e andamos em direção ao último segmento procurando segmentos que sejam maiores que o limiar de energia, quando encontramos, analisamos para ver se pelo menos 3 dos 5 segmentos subseqüentes estão acima do limiar de energia. Caso estejam, guardamos este primeiro segmento como o segmento inicial de uma possível palavra e continuamos a procura até encontrarmos um possível final para a palavra que também será testado para ver se pelo menos 3 dos 5 segmentos subseqüentes estão abaixo do limiar de energia. Este procedimento é repetido por todo o sinal de fala até que no final tenhamos um conjunto de possíveis segmentos iniciais e finais de várias possíveis palavras. Depois fazemos uma procura nos segmentos anteriores aos possíveis segmentos inicias e nos subseqüentes aos possíveis segmentos finais encontrados por energia. Esta procura é feita através de um limiar de cruzamentos por zero, onde analisamos se há pelo menos 3 dos 5 segmentos anteriores ou subseqüentes que satisfaçam a condição de limiar. Caso seja verdadeira a resposta, então assumimos o último a cruzar o limiar de cruzamentos por zero como o novo possível início ou fim e analisamos os próximos 5 segmentos até que a resposta não seja mais verdadeira.

Agora já teremos novos possíveis segmentos iniciais e finais encontrados pelos cruzamentos por zero. É neste ponto que analisamos se as várias possíveis palavras podem se juntar, formando um conjunto menor de palavras. Neste procedimento, analisamos quantos segmentos separam cada início e fim de palavras subseqüentes. Caso o número de segmentos seja menor que 3, juntamos estas duas possíveis palavras em uma única, uma vez que, fizemos todas as possíveis junções. Então analisamos o tamanho das possíveis palavras que separamos; e as que possuírem tamanho menor que 10 segmentos serão desconsideradas. Como o objetivo é separar uma única palavra, então ficamos sempre com a de maior tamanho.

Passos para execução do algoritmo:

1) Estipulamos os limiares de energia e de cruzamentos por zero.
2) Começamos a procura de cruzamento do sinal pelo limiar de energia. Ao encontrarmos, fazemos um teste para ver se pelo menos três dos próximos cinco segmentos estão acima do limiar de energia, caso estejam, assumimos este segmentos como o segmento inicial desta possível palavra e continuamos a procura por um outro cruzamento do sinal com o limiar de energia, e ao encontrarmos fazemos o teste para ver se pelo menos três dos próximos cinco segmentos estão abaixo do limiar de energia, caso estejam, este segmento será assumido como fim da possível palavra. Quando não ocorrer pelo menos três dos cinco segmentos testados que satisfaçam a condição, a procura continua e não assumimos nada sobre este segmento. Fazemos isto para todo sinal, encontrando início e fim de possíveis palavras dentro do sinal.
3) Apartir destes pontos procuramos cruzamento do sinal com limiares de cruzamentos por zero, da mesma forma que será feito no algoritmo #2, porém com as mesmas condições de teste e procura apresentadas no item 2 acima.
4) Temos agora várias possíveis palavras já recortadas tanto por energia como por cruzamentos por zero, então testamos para ver se a distância entre as palavras são suficientemente grandes (maior que três segmentos), ou seja, se temos realmente mais de uma palavra ou se as palavras que separamos nada mais são que pedaços de uma única palavra.
5) Agora analisamos o tamanho das palavras recortadas, e se alguma tiver tamanho menor ou igual a dez segmentos, esta será descartada.
6) No final quando temos recortadas mais de uma palavra, escolhemos a de tamanho maior, pois, estamos interessados em obter somente uma palavra recortada por sinal analisado.
7) Fim do algoritmo.

Os limiares tanto de energia quanto de cruzamentos por zero foram propostos empiricamente após análises dos limiares tanto ouvindo quanto observando os gráficos de energia e cruzamentos por zero por segmento, então estipulamos o limiar de energia como 10x(média de energia dos 20 primeiros segmentos) e o limiar de cruzamentos por zero como 3x(média de

cruzamentos por zero dos 20 primeiros segmentos). De forma análoga ao algoritmo proposto por Rabiner e Sabur(1979), os 20 primeiros segmentos é a gravação do ruído de fundo do local onde ocorreu a gravação.

Devido a não utilização de todas as informações estatísticas do ruído de fundo, veremos que este algoritmo não tem um resultado tão bom quanto o algoritmo #2, proposto por Rabiner e Sabur [3]. Estes resultados serão analisados na próxima seção.

2.3.2 – Algoritmo #2 (Algoritmo proposto pelo Rabiner e Sabur [3])

Agora, abordaremos uma técnica bastante simples utilizada na detecção de extremos. Este método que iremos descrever foi publicado por Rabiner e Sambur (1975). Primeiramente, ao gravarmos o sinal deixamos os 10 primeiros segmentos do sinal sem nenhum sinal de fala, somente com ruído de fundo do ambiente. Apartir dos resultados da energia e dos cruzamentos por zero destes 10 primeiros segmentos, encontramos as estatísticas do ruído de fundo do sinal com as quais geramos os limiares superior e inferior para a energia, que são respectivamente τ_u e τ_l. Para encontrarmos os primeiros prováveis extremos do sinal é necessário que nos movamos simultaneamente dos dois extremos do sinal em direção ao meio, procurando o primeiro cruzamento do sinal com o limiar superior. Uma vez encontrado estes dois pontos (inicial e final para um possível recorte no sinal), nós devemos voltar em direção aos extremos do sinal procurando um possível cruzamento com limiar inferior, de modo que estes novos valores N_1 e N_2 sejam os limiares para o recorte do sinal encontrados por energia. Apartir de N_1 e N_2, andamos em direção aos extremos do sinal, não mais de 25 segmentos, para ver se encontramos pelo menos 3 ocorrências acima do limiar de cruzamentos por zero τ_{zc}. Se estas ocorrências forem encontradas, então o novo limiar será a ocorrência mais distante de N_1 ou N_2 acima do limiar τ_{zc} ,ou seja, $\hat{N}_1$ ou $\hat{N}_2$.

Passos para execução do algoritmo:

1) Estipular valores para os limiares (τ_u, τ_l e τ_{zc}).
2) Andar simultaneamente do início e do fim do sinal procurando o primeiro cruzamento com o limiar superior de energia. Encontramos dois pontos, um mais próximo do início e outro mais próximo do fim.
3) Apartir dos pontos encontrados acima, andamos simultaneamente em direção ao início e ao fim do sinal, procurando o primeiro cruzamento entre o sinal e limiar inferior de energia, os segmentos onde ocorrem o cruzamento são respetivamente N_1 e N_2.
4) Apartir de N_1 andamos em direção ao início, não mais que 25 segmentos procurando pelo menos três cruzamentos pelo limiar de cruzamentos por zero, caso ocorra pelo menos três cruzamentos, o novo recorte será dado pelo segmento mais distante de N_1 (em direção ao início do sinal) que cruzou o limiar de cruzamentos por zero. O mesmo procedimento é feito para N_2 em direção ao fim do sinal, ao final destes dois procedimentos

encontramos $\hat{N}_1$ e $\hat{N}_2$. Quando não ocorre pelo menos três cruzamentos pelo limiar de cruzamentos por zero, então o limiar $\hat{N}_i$ será igual a N_i.

5) Fim do algoritmo.

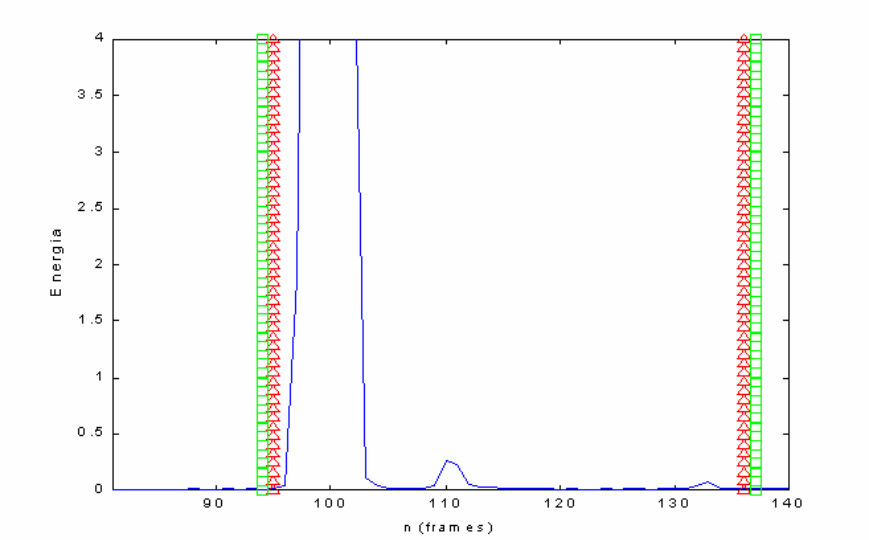


Figura 12: Representa os recortes feitos através dos dois limiares de energia. Os triângulos (vermelho) representam os cortes realizados através dos limiares superiores de energia e os quadrados (verde) representam os cortes referente ao limiar inferior de energia.

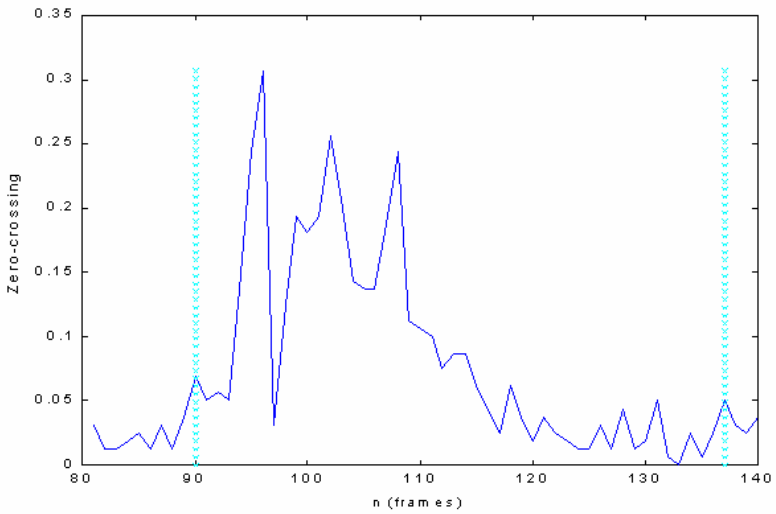


Figura 13: Representa os recortes feitos através do limiar de cruzamentos por zero.

As figuras 12 e 13 representam os recortes feitos na palavra “faca”, podemos assim notar que o limiar de energia mais externo que cruza o sinal entre os segmentos de número 130 e 140, no caso N_2, não se alterou com a procura de cruzamentos com o limiar de cruzamentos por zero. Porém, o limiar

de energia mais externo que cruza o sinal entre os segmentos número 90 e 100, no caso N_1, foi alterado devido a procura de cruzamentos com limiar de cruzamentos por zero que ocorreu com o segmento de número 90. Isto já era previsto, pois, o sinal que fora recortado representa a palavra “faca” que tem uma parte não vozeada no início(“f”) e outra vozeada no fim(“a”) - não estão em análise as características das partes internas do sinal, somente o início e o fim do sinal.

2.4 – Resultados comparativos dos dois algoritmos

A forma de se analisar os resultados tem um certo grau de subjetividade, tendo em vista que nem sempre somos capazes de concluir com precisão o ponto certo onde o sinal deve ser recortado, de modo que se elimine somente o ruído de fundo do sinal.

De posse dos resultados dos algoritmos #1 e #2, constatei que este primeiro gerou resultados bem melhores que o segundo, pois, no #1 ocorreram capturas de sinais que não tinham informação alguma, e no #2 mesmo usando diferentes valores multiplicativos nos limiares, sempre conseguimos obter resultados bastante razoáveis. Todos o sinais recortados sempre continham boa parte da informação de fala do sinal, não ocorrendo quase nunca um recorte de uma parte onde não houvesse nenhuma informação. Evidentemente, a primeira idéia que nos passa pela cabeça é de modificar os limiares do algoritmo #1 mas, nenhuma das mudanças promovidas nestes limiares, geraram resultados suficientemente satisfatórios para que este pudesse ser considerado comparativamente mais eficiente que o algoritmo ou pelo menos equivalente em matéria de eficiência.

Adiante temos as tabelas de resultados para o algoritmo #2, onde o conjunto de teste eram os 5 sinais de cada classe, sendo as classes os dígitos de 0 a 9, logo no total teríamos 50 sinais para o teste. Neste caso, **limiares: axb**, significa que o valor de **a** é o fator multiplicativo do limiar de energia e **b** é o fator multiplicativo do limiar de cruzamentos por zero. Foram assumidos como erros os recortes onde ocorrem perdas de informação e dúvidas quando pegamos ruído além do sinal de fala, não ocorrendo perda de informação.

Os limiares do algoritmo #1 são dados por:

- **Limiar de energia** = **a**x(média da energia do ruído de ambiente) .
- **Limiar de cruzamentos por zero** = **b**x(média do cruzamentos por zero do ruído de ambiente).

Tomemos por base a tabela 2, que por sinal foi a que nos proporcionou melhor resultado. Então, alteramos na tabela 3 o limiar de cruzamentos por zero, no caso aumentamos e como era de se esperar, o número de casos de erro aumentaram, pois com o aumento do limiar deixamos de pegar segmentos não vozeados que com o limiar anterior eram pegos. Por conseguinte, recortando o sinal ainda dentro do sinal de fala e perdendo informação

importante. Já na tabela 4, aumentamos o limiar de energia e mantivemos o mesmo limiar de cruzamentos por zero da tabela 2, observando assim um pequeno aumento no número de erros, pois, os recortes relativos aos limiares de energia foram feitos mais internamente, ou seja, possibilitando a perda de informação de fala, contudo, graças ao limiar de cruzamentos por zero que é sempre testado mais externamente aos limiares de energia, os números de recortes errados não aumentaram tanto.

Tabela 2: Resultados do recorte (Algoritmo #1).

	Algoritmo #1 - Dígitos - limiares:10x3									
	1	2	3	4	5	6	7	8	9	0
Acertos	4	5	5	0	0	2	3	0	0	5
Erros	0	0	0	4	3	0	1	0	0	0
Dúvidas	1+	0	0	1+	2+	3+	1+	5+	5+	0

	Total (%)
Acertos	48
Erros	16
Dúvidas	36

Tabela 3: Resultados do recorte (Algoritmo #1).

	Algoritmo #1 - Dígitos - limiares:10x7									
	1	2	3	4	5	6	7	8	9	0
Acertos	4	4	4	0	0	2	2	0	1	5
Erros	0	0	0	4	5	2	3	3	0	0
Dúvidas	1+	1+	1+	1+	0	1+	0	2+	4+	0

	Total (%)
Acertos	44
Erros	34
Dúvidas	22

Tabela 4: Resultados do recorte (Algoritmo #1).

	Algoritmo #1 - Dígitos - limiares:15x3									
	1	2	3	4	5	6	7	8	9	0
Acertos	4	5	4	0	0	0	0	0	5	5
Erros	0	0	0	4	3	2	1	0	0	0
Dúvidas	1+	0	1+	1+	2+	3+	4+	5+	5+	0

	Total (%)
Acertos	42
Erros	22
Dúvidas	36

Adiante analisaremos os resultados provenientes do recorte feito pelo algoritmo #2. Os limiares: **axbxc**, significam respectivamente os fatores multiplicativos do limiar superior de energia, do limiar inferior de energia e do limiar de cruzamentos por zero.

Os limiares do algoritmo #2 são dados por:

- **Limiar superior de energia** = (média da energia do ruído de ambiente) + **a**x(variância da energia do ruído ambiente) .
- **Limiar inferior de energia** = (média da energia do ruído de ambiente) + **b**x(variância da energia do ruído ambiente) .
- **Limiar de cruzamentos por zero** = (média dos cruzamentos por zero do ruído de ambiente) + **c**x(variância dos cruzamentos por zero do ruído ambiente) .

Tomando por base os resultados da tabela 5, notamos que na tabela 6 houve um aumento no número de recortes errados devido a um aumento no limiar de cruzamentos por zero. Já na tabela 7 houve um aumento no número de recortes duvidosos, pois, diminuímos o limiar inferior de energia de modo a recortarmos o sinal mais externamente, capturando níveis mais baixos de energia e consequentemente capturando mais ruído de ambiente tanto antes quanto depois do sinal de fala.

Vale a pena ressaltar que com estes limiares da tabela 7, nós não obtivemos nenhum erro de recorte. Então aumentamos um pouco o limiar de energia inferior para tentarmos aumentar o número de acertos e pelo menos mantermos o número de erros que no caso da tabela 7 é zero. Este resultado conseguimos com a tabela 8, onde obtivemos um bom número de acertos e nenhum erro.

Tabela 5: Resultados do recorte (Algoritmo #2).

	Algoritmo #2 - Dígitos - limiares:10x6x3									
	1	2	3	4	5	6	7	8	9	0
Acertos	5	4	5	4	4	1	3	4	4	5
Erros	0	0	0	0	0	1	0	0	0	0
Dúvidas	0	1+	0	1+	1+	3+	2+	1+	1+	0

	Total (%)
Acertos	78
Erros	2
Dúvidas	20

Tabela 6: Resultados do recorte (Algoritmo #2).

	Algoritmo #2 - Dígitos - limiares:10x6x6									
	1	2	3	4	5	6	7	8	9	0
Acertos	5	5	3	2	5	2	2	3	5	2
Erros	0	0	0	0	0	1	0	0	0	3
Dúvidas	0	0	2+	3+	0	2-	3+	2+	0	0

	Total (%)
Acertos	70
Erros	8
Dúvidas	22

Tabela 7: Resultados do recorte (Algoritmo #2).

Algoritmo #2 - Dígitos - limiares:10x4x3										
	1	2	3	4	5	6	7	8	9	0
Acertos	5	0	4	4	3	1	1	3	4	5
Erros	0	0	0	0	0	0	0	0	0	0
Dúvidas	0	5+	1+	1+	2+	4+	4+	2+	1+	0

	Total (%)
Acertos	60
Erros	0
Dúvidas	40

Tabela 8: Resultados do recorte (Algoritmo #2).

Algoritmo #2 - Dígitos - limiares:10x5x3										
	1	2	3	4	5	6	7	8	9	0
Acertos	5	4	5	4	4	2	3	4	4	5
Erros	0	0	0	0	0	0	0	0	0	0
Dúvidas	0	1+	0	1+	1+	3+	2+	1+	1+	0

	Total (%)
Acertos	80
Erros	0
Dúvidas	20

Comparando os resultados das tabela que usam o algoritmo #1 e as que usam o algoritmo #2, notamos resultados muito melhores nas que usam o algoritmo #2, porém entre si os resultados não variam muito quando alteramos os limiares. Também já era de se esperar que os resultados do algoritmo proposto pelo Rabiner [3] (algoritmo #2) fosse superior ao algoritmo #1, pois, o primeiro usa duas informações estatísticas do sinal, que são respectivamente a média da energia e cruzamentos por zero do ruído ambiente e a variância da energia e do cruzamentos por zero do mesmo. O algoritmo #1 usa somente a média de energia e cruzamentos por zero do ruído ambiente. Deste modo, sendo menos independente dos fatores multiplicativos dos limiares que o primeiro e por conseguinte menos eficiente.

3 – Extração de parâmetros

O bloco de extração de parâmetros é de importância vital para o nosso sistema de reconhecimento, pois, a operação básica de um sistema de reconhecimento é comparar segmentos de fala; sejam eles: segmentos, fonemas ou qualquer outra unidade de fala; e gerar um resultado do quão parecidas estas unidades são (medida de similaridade).

A idéia por trás da extração de parâmetros é representar segmentos de sinais de fala com o menor número possível de parâmetros, de modo que estes carreguem “toda” a informação necessária do sinal. Entende-se por “toda” informação necessária, o mínimo de informação suficiente para caracterizar o sinal. Estas características são dadas pela forma das cavidades do trato vocal, inclusive os lábios.

Como já fora mencionado, o modelo de produção da fala é dado por uma excitação que pode ser periódica ou ruidosa; o que significa respectivamente som vozeado e não vozeado. De forma não muito criteriosa podemos dizer que: sons vozeados definem regiões de ressonância, e não vozeados definem regiões de anti-ressonância, e estas freqüências de ressonância são devido a formação de várias cavidades e sub-cavidades acústicas no processo de produção da fala. Cada cavidade e sub-cavidade terá uma freqüência de ressonância que só dependerá da sua forma e comprimento. As freqüências nominais centrais destas ressonâncias são chamadas de formantes. Deste modo, fica claro que um sinal de fala terá vários formantes. Teoricamente este número seria infinito, contudo, na prática geralmente encontramos de 3 a 5 na banda de Nyquist. O primeiro formante é o de freqüência mais baixa, o segundo é o de segunda freqüência mais baixa e assim por diante.

Neste trabalho nós apresentaremos dois modelos de extração de parâmetros, um baseado em análise LPC (Linear Predictive Coding) e outro na análise cepstral. Os parâmetros extraídos nestas análises são os próprios coeficientes do modelo utilizados para representar o sinal de fala. Estes modelos não tratam exatamente de codificação no sentido de transformar uma forma de onda em uma seqüência de números e depois recuperar esta forma de onda original pois, não é este o nosso objetivo. O que significa que não veremos quantizadores e nem a parte de síntese que seriam fundamentais para a codificação no sentido mencionado acima.

3.1 – Análise LP (Linear Prediction)

Para iniciarmos a análise LP, precisamos primeiramente apresentar o modelo discreto no tempo para a produção da fala. Este é baseado num filtro que simula as características das cavidades acústicas mais o tipo de excitação relativa ao tipo de som(vozeado/não vozeado).

3.1.1 – Modelo “ Detalhado” x “Simplificado”

Abaixo, mostraremos um diagrama deste modelo e suas equações, sendo que separaremos o modelo “detalhado” e o estimado pela análise LP, modelo “simplificado”.

$$\Theta(z) = \Theta'_0 \cdot \frac{1 + \sum_{i=1}^{L} b(i) \cdot z^{-1}}{1 + \sum_{i=1}^{N} a(i) \cdot z^{-1}}, \text{modelo "detalhado".}$$

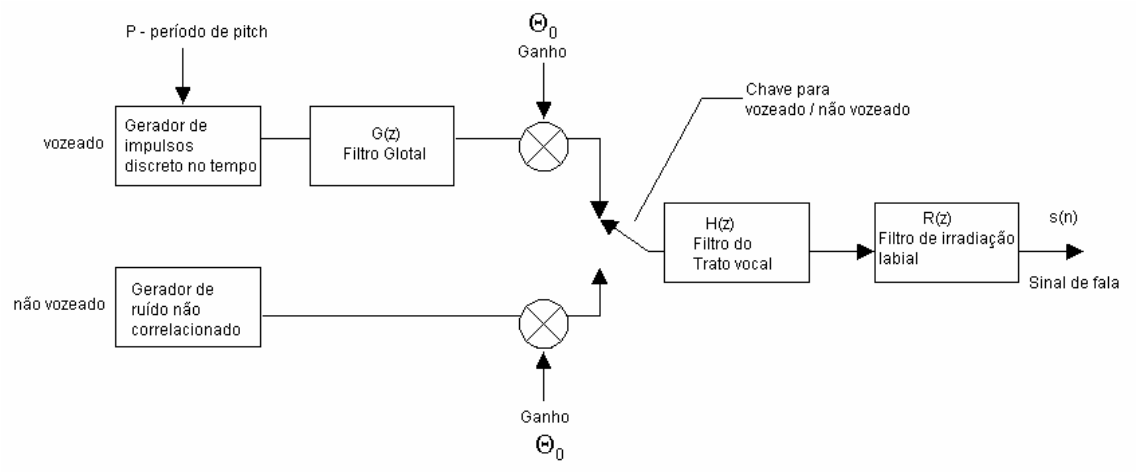


Figura 14: Modelo “detalhado” da produção de sinal de fala discreto no tempo.

$$\Theta(z) = \Theta'_0 \cdot \frac{1}{1 + \sum_{i=1}^{N} \hat{a}(i) \cdot z^{-1}}, \text{modelo "simplificado".}$$

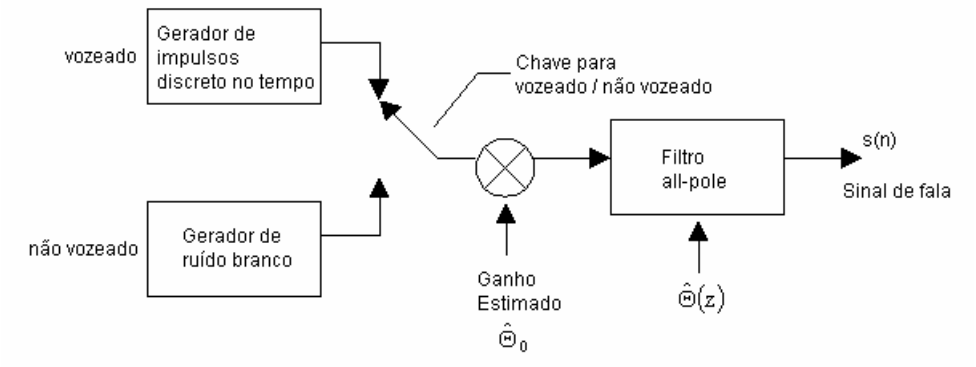


Figura 15: Modelo “simplificado” por predição linear.

Neste momento vale a pena discutirmos porquê o modelo de pólos e zeros para produção da fala pode ser simplificado pelo modelo só de pólos. A

idéia de se ter pólos e zeros no modelo é bastante razoável e natural pois, se pensarmos em sons como os das vogais podemos perceber que seu espectro aproximadamente periódico pode ser representado por um filtro só de pólos(all-pole). Já ao observarmos os sons nasais e fricativos, notamos pontos nulos no espectro; o que pode ser representado matematicamente através da inclusão de zeros no filtro.

A utilização de um modelo all-pole é primeiramente uma necessidade matemática, pois, este pode ser facilmente resolvido através de equações lineares, ou seja, para encontrarmos os coeficientes do filtro utilizaremos equações lineares simples, sendo assim, um modelo fácil e eficiente computacionalmente.

Além da necessidade, existe um outro motivo bastante importante para a utilização do modelo só de pólos, e para entendermos, precisamos ter em mente o objetivo do modelo LP: preservar a informação acústica sem se preocupar com possíveis distorções temporais. Esta proposição do modelo LP fica clara quando percebemos que a informação de um sinal de fala independe da fase do sinal. Como exemplo, podemos citar o caso clássico de um orador parado e um ouvinte andando por entre os cômodos de uma casa, é evidente que a fase muda conforme ele anda e mesmo assim ele consegue entender toda a informação transmitida pelo orador; caso esta tenha potência suficiente para chegar ao ouvinte com amplitude mínima audível.

Pelo que expomos acima, fica claro que o modelo pode ser representado desconsiderando-se a fase e considerando a magnitude do sinal. Um tipo de filtro que atende bem estas especificações é o filtro só de polos (*“all-pole”*). Este modelo será capaz de representar corretamente a magnitude do espectro, porém, com fase mínima relativa ao modelo “verdadeiro”. Como a intenção é de armazenar e transportar só a informação acústica, este modelo é perfeitamente válido e adequado as nossas necessidades.

A seguir mostraremos técnicas utilizadas na obtenção dos coeficientes de predição linear, sendo elas: pré-ênfase do sinal de fala, método para extração de parâmetros e medidas de distância entre sinais representados por estes parâmetros.

3.1.2 – Pré-ênfase

O filtro de pré-ênfase deve ser aplicado ao sinal de fala original antes de passá-lo por qualquer outro processamento. Este filtro é da forma:

$$P(z) = 1 - \mu z^{-1}, \text{ onde } 0 \le \mu \le 1.$$

O valor de μ é mais ou menos estabelecido ao redor de 0.95 e o valor utilizado em nosso trabalho foi 0.97 por ser um valor muito comum em publicações internacionais.

A pré-ênfase aumenta a energia relativa do espectro de alta freqüência e as razões para sua utilização é devido a prevenção contra instabilidade numérica, pois, como veremos futuramente no método da autocorrelação para a obtenção dos coeficientes dos filtros, sendo um sinal de fala basicamente composto por componentes de baixa freqüência. Caso utilizarmos um grande número de coeficientes LP para representar este sinal, poderemos incorrer numa matriz de autocorrelação mal condicionada. Se o espectro tem um "tilt" que ocasiona esta larga faixa dinâmica, um filtro de primeira ordem é capaz de torná-lo mais parecido com o espectro do ruído branco, cuja matriz de autocorrelação é bem condicionada (de fácil inversão).

3.1.3 – Método para calcular os parâmetros

Apresentaremos abaixo, o método da autocorrelação para extração de parâmetros. O modelo de predição linear apresentado no diagrama da seção 3.1.1 pode ser representado pela equação:

$$\hat{S}(z) = \hat{\Theta}(z) \cdot E(z)$$

onde:

$$\hat{\Theta}(z) = \frac{1}{\hat{A}(z)}$$

$$\hat{A}(z) = a(0) + \sum_{i=1}^{M} a(i) \cdot z^{-1} = 1 - \sum_{i=1}^{M} \hat{a}(i) \cdot z^{-1}$$

nada mais é que a transformada Z do sinal de fala estimado pelo modelo LP. Através desta equação podemos chegar a equação no domínio do tempo que será dada por:

$$\hat{s}(n) = \sum_{i=1}^{M} \hat{a}(i) \cdot \hat{s}(n-i) + \hat{\Theta}_0 \cdot e(n)$$

sendo:

$\hat{a}(i) =>$ coeficientes ou parâmetros de predição linear.

$\hat{\Theta}_0 =>$ uma constante relacionada Θ_0 (do modelo verdadeiro).

$e(n) =>$ excitação do modelo.

Fica evidente com o modelo no domínio do tempo que s(n) pode ser obtido através de uma combinação linear de seus i valores passados, daí surgiu nome de predição linear, pois se tivermos de posse dos i valores passados poderemos estimar um valor atual/futuro de s. Em termos estatísticos este modelo é conhecido como autoregressivo (AR).

Agora analisando o erro de predição que fica evidenciado no diagrama a seguir, poderemos obter resultados que serão de bastante utilidade no cálculo

dos parâmetros. Primeiro assumiremos que $\hat{s}(n) = \sum_{i=1}^{M} \hat{a}(i) \cdot s(n-i)$, onde $\hat{s}(n)$ é predição de s(n); $P(z) = \sum_{i=1}^{M} \hat{a}(i) \cdot z^{-1}$, onde P(z) é um filtro de predição FIR; e ê(n) é o erro predição.

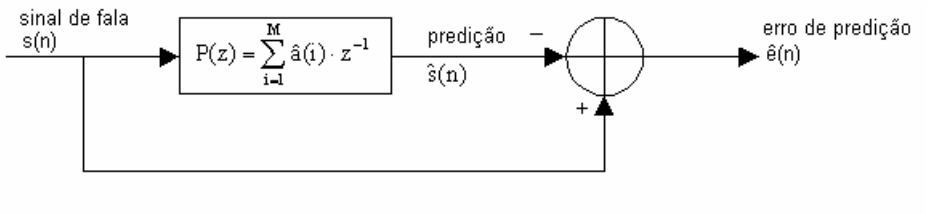


Figura 16: Diagrama do erro de predição.

Então a equação do erro de predição será dada por:

$$\hat{e}(n) = s(n) - \hat{s}(n) = s(n) - \sum_{i=1}^{M} \hat{a}(i) \cdot s(n-i).$$

O melhor valor de $\hat{s}(n)$ é quando este valor está mais próximo de s(n) no sentido médio quadrático, o que significa minimizar o erro de predição. Este resultado é alcançado derivando-se o erro médio quadrático (MSE) e igualando-o a zero.

$$\frac{\partial(E\{[\hat{e}(n)]^2\})}{\partial a_\alpha} = E\left\{2 \cdot \left[s(n) - \left(\sum_{i=0}^{M} \hat{a}(i) s(n-i)\right)\right] \cdot s(n-\alpha)\right\} = 0$$

$$\text{podemos reescrever :}$$

$$E\{s(n) \cdot s(n-\alpha)\} - E\left\{\sum_{i=1}^{M} \hat{a}(i) \cdot s(n-i) \cdot s(n-\alpha)\right\} = 0$$

Sabendo que a definição de autocorrelação é:

$$r_s(k) = E\{s(n) \cdot s(n-k)\}, \text{deste modo } r_s \text{ só depende de k.}$$

A equação desenvolvida pelo MSE pode ser rescrita como:

$$\text{Forma analítica :}$$

$$r_s(\alpha) = \sum_{i=1}^{M} \hat{a}(i) \cdot r_s(\alpha - i), \text{sendo } \alpha = 1, 2, ..., M.$$

$$\text{Forma Matricial :}$$

$$\begin{bmatrix} r(0) & r(1) & r(2) & \cdots & r(M-1) \\ r(1) & r(0) & r(1) & \cdots & r(M-2) \\ r(2) & r(1) & r(0) & \cdots & r(M-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r(M-1) & r(M-2) & r(M-3) & \cdots & r(0) \end{bmatrix} \cdot \begin{bmatrix} \hat{a}(1) \\ \hat{a}(2) \\ \hat{a}(3) \\ \vdots \\ \hat{a}(M) \end{bmatrix} = \begin{bmatrix} r(1) \\ r(2) \\ r(3) \\ \vdots \\ r(M) \end{bmatrix}.$$

Tomamos a liberdade de não usar o índice s na forma matricial pois, já sabemos que este é só uma referencia ao sinal s(n). Escrevendo simplificadamente, $\underline{R} \cdot \underline{\hat{a}} = \underline{r}$, onde $\underline{R}$ é a matriz de autocorrelação, $\underline{\hat{a}}$ é o vetor de parâmetros de predição linear e $\underline{r}$ é o vetor de autocorrelação.

Apartir deste resultado podemos concluir que só dependemos do próprio sinal s(n) para obtermos os parâmetros â(i) do modelo de predição linear. Veremos mais claramente na equação abaixo:

$\underline{R} \cdot \underline{\hat{a}} = \underline{r}$, de onde podemos concluir que $\underline{\hat{a}} = \underline{R}^{-1} \cdot \underline{r}$

Antes de continuarmos, é oportuno relembrar que todas estas operações são feitas em cada segmento do sinal de fala pois, desde o início do trabalho assumimos que segmentos pequenos de sinais de fala podem ser considerados aproximadamente estacionários, o que nos permitiu fazer uma série de considerações matemáticas que facilitou nosso trabalho. Concluindo, para cada segmento teremos um vetor de parâmetros â(i), onde i será escolhido de acordo com o nosso interesse.

Um método eficiente de se resolver a equação acima onde obtemos os parâmetros através das matrizes $\underline{R}$ e $\underline{r}$ é usar o algoritmo de recursão Levinson-Durbin, que será descrito a seguir.

Algoritmo:

1º passo - Inicialização:

$\xi^0 = r(0) \Rightarrow$ energia do sinal s(n).

2º passo - Recursão para i=1, 2, …, M:

$$k(i) = \frac{\left(r(i) - \sum_{j=1}^{i-1} \hat{a}^{i-1}(j) \cdot r(i-j) \right)}{\xi^{i-1}}; \text{ coeficientes de reflexão}$$

$\hat{a}^i(i) = k(i);$

$\hat{a}^i(j) = \hat{a}^{i-1}(j) - k(i) \cdot \hat{a}^{i-1}(i-j); \text{ para } 1 \le j \le i$

$\xi^i = (1 - k^2(i)) \cdot \xi^{i-1};$

3º passo - Fim.

Para realizarmos este algoritmo é preciso ter em mãos os valores do vetor de autocorrelação $\underline{r}$ que é facilmente obtido pela própria definição de autocorrelação. Numericamente pode ser escrito como:

$$r(k) = \frac{1}{L} \cdot \sum_{n=k+1}^{L} s(n) \cdot s(n-k),$$

assumindo que L é o tamanho do segmento e n=1, 2, …,L.

3.2 – Medidas de distância

São medidas utilizadas para comparar um conjunto de parâmetros com outro conjunto de mesma dimensão. No caso de sinais de fala, nós podemos comparar os **p** parâmetros de um segmento **x** com os **p** parâmetros de um segmento **y**. Quanto menor o valor da distância mais próximo (parecido) os segmentos serão. Chamaremos de $\hat{a}(i)$ e $\hat{b}(i)$ os vetores de parâmetros LP de dois segmentos diferentes e fazendo $1 \le i \le p$. Mostraremos abaixo algumas das medidas mais comuns utilizadas.

- Distância Euclidiana:
  É a medida mais básica de distância. Se assumirmos **p=2**, esta representará a distância geométrica entre dois pontos no plano cartesiano. Deste modo, podemos generalizar este conceito para um espaço de dimensão **N**, fazendo **p=N**.

$$d(\underline{\hat{a}}, \underline{\hat{b}}) \overset{def}{=} \sqrt{\sum_{k=1}^{p} \left(\hat{a}(k) - \hat{b}(k)\right)^2}\,.$$

- Distância de Itakura:
  Mede o quão melhor $\underline{\hat{a}}(i)$ consegue predizer seu próprio segmento em relação ao $\hat{b}(i)$. Simplificando, seria o logaritmo da relação entre os erros de predição de $\underline{\hat{a}}(i)$ predizendo o próprio segmento, e $\hat{b}(i)$ predizendo o segmento relativo aos coeficientes de $\underline{\hat{a}}(i)$.

$$\underline{\tilde{R}}_s \overset{def}{=} \begin{bmatrix} r_s(0) & \underline{r}_s^{\,T} \\ \underline{r}_s & \underline{R}_s \end{bmatrix}, \text{ sendo o frame realcionado aos parâmetros } \underline{\hat{a}}(i)$$

$$\underline{\alpha} = \begin{bmatrix} 1 & -\underline{\hat{a}}^T \end{bmatrix},$$

$$\underline{\beta} = \begin{bmatrix} 1 & -\underline{\hat{b}}^T \end{bmatrix},$$

$$d_I(\underline{\hat{a}}, \underline{\hat{b}}) \overset{def}{=} \log \frac{\xi_{\underline{\hat{b}}}}{\xi_{\underline{\hat{a}}}} = \log \frac{\underline{\beta}^T \cdot \underline{\tilde{R}}_s \cdot \underline{\beta}}{\underline{\alpha}^T \cdot \underline{\tilde{R}}_s \cdot \underline{\alpha}}$$

  Vale a pena ressaltar que $d_I(\underline{\hat{a}}, \underline{\hat{b}}) \neq d_I(\underline{\hat{b}}, \underline{\hat{a}})$ e $d(\underline{\hat{a}}, \underline{\hat{b}}) = d(\underline{\hat{b}}, \underline{\hat{a}})$, no caso da distância de Itakura, o resultado depende se estamos comparando $\underline{\hat{a}}$ com $\underline{\hat{b}}$ ou $\underline{\hat{b}}$ com $\underline{\hat{a}}$. Já no caso da distância Euclidiana, independe da ordem dos termos a serem comparados.

3.3 – Análise Cepstral

Primeiramente iremos expor de onde surgiu a idéia da análise cepstral. Relembrando que o modelo utilizado de sinal de fala é caracterizado pela convolução do sinal de excitação com a resposta ao impulso do modelo do trato vocal.

Esta modelagem nos gera algumas limitações pois, não nos possibilita a separação das duas componentes, só nos fornecendo o resultado (saída do sistema). Como o nosso intuito é representar os segmentos de sinais de fala, seria de grande utilidade separarmos a excitação do modelo do trato vocal, porque este é o que caracteriza o ambiente de formação da fala.

A representação no domínio da freqüência (espectro) ainda não nos possibilitaria a separação destas duas componentes do sinal de fala, pois, teríamos uma multiplicação; o quê não seria fácil de se separar. Daí surgiu a idéia de se criar um novo domínio onde fôssemos capazes de representar o sinal de fala como uma combinação linear da excitação com a resposta ao impulso do modelo do trato vocal já que, a engenharia nos fornece uma série de ferramentas que nos auxilia na separação de componentes combinados linearmente.

Um sinal de fala s(n) é dado pela seguinte equação:
$s(n) = e(n) * \theta(n)$, (convolução)
onde
s(n) é o sinal de fala.
e(n) é o sinal de excitação.
$\theta(n)$ é a resposta ao impulso do modelo do trato vocal.

Assim como o espectro (em inglês *"spectrum"*), o domínio cepstral ou simplesmente cepstrum representa a transformação do sinal de fala com duas propriedades importantes:
1ª - As componentes do sinal, e(n) e $\theta(n)$, serão separadas no cepstrum.
2ª - As componentes do sinal, e(n) e $\theta(n)$, serão linearmente combinadas no cepstrum.

O domínio cepstral é separado em dois: o cepstrum real - RC (real cepstrum); e o cepstrum complexo – CC (complex cepstrum). A diferença básica entre os dois é o RC descarta a informação de fase do sinal, ou seja, o sinal é sempre representado como sendo de fase mínima e o CC considera a informação de fase.

Notações:

Domínio	Notação para um sinal x(n) (no tempo)
CC	$\gamma_x(n)$

RC	$c_x(n)$

3.3.1 – Cepstrum real

A definição de cepstrum real nos diz que:

$$c_s(n) \overset{def}{=} \Im^{-1}\left\{\log\left|\Im\{s(n)\}\right|\right\} = \frac{1}{2\cdot\pi}\int_{-\pi}^{\pi}\log\left|S(\omega)\right|\cdot e^{j\omega n}\cdot\partial\omega.$$

Ao analisarmos o que significa estas operações podemos concluir que as separação e combinação linear dos componentes que representam o sinal, e(n) e $\theta(n)$, serão satisfeitas.

Analisando:

Sinal de fala (no tempo) (Fourier) Espectro de s(n) (na freqüência)

$$s(n) = e(n) * \theta(n) \xrightarrow{\Im} S(\omega) = E(\omega)\cdot\Theta(\omega)$$

Aplicando o logaritmo em $S(\omega)$, teremos:

$$\log\{S(\omega)\} = \log\{E(\omega)\cdot\Theta(\omega)\} = \log\{E(\omega)\} + \log\{\Theta(\omega)\} = C_s(\omega);$$

(Inversa de Fourier)

$$C_s(\omega) = C_e(\omega) + C_\theta(\omega) \xrightarrow{\Im^{-1}} c_s(n) = c_e(n) + c_\theta(n)$$

redefinindo:

Cepstrum de s(n) (na quefrência)

$$c_s(n) = c_e(n) + c_\theta(n).$$

Observamos que primeiramente fazemos a transformada de Fourier do sinal de fala s(n). Deste modo, obtendo uma multiplicação das componentes e(n) e $\theta(n)$ na freqüência e depois aplicando o operador logarítmico no espectro de s(n), ou seja, no final teremos que o logaritmo do espectro de s(n) será igual a soma dos logaritmos do espectro de e(n) e de $\theta(n)$. Para finalizar só falta aplicarmos a transformada inversa de Fourier, assim, obteremos o cepstrum de s(n). Comparando o cepstrum com o espectro, o que equivaleria a freqüência no espectro será chamado de qüefrência no cepstrum. Resumindo, s(n) no domínio da qüefrência será igual a soma da excitação e(n) no domínio da qüefrência e da resposta ao impulso do modelo do trato vocal $\theta(n)$ no domínio da qüefrência.

Abaixo mostraremos como fica o diagrama de blocos do sinal s(n) segmentado.

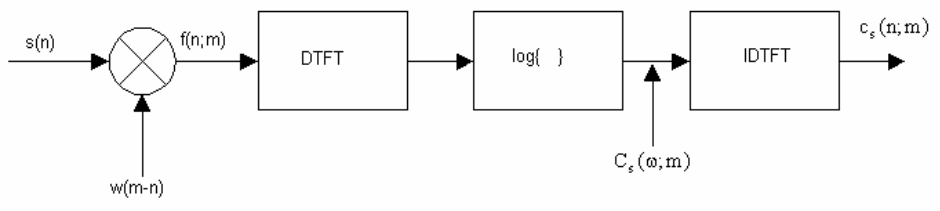


Figura 17: Diagrama de blocos da representação cepstral do sinal s(n).

Uma vez de posse de uma representação do sinal s(n) num domínio onde podemos separar e(n) e $\theta(n)$, basta utilizarmos algum processo linear para extrair a informação que nos interessa de $c_s(n)$. Como a informação que nos interessa é referente a resposta ao impulso do modelo do trato vocal, podemos retirar a informação relativa ao sinal de excitação $c_e(n)$ e ficaremos somente com $c_\theta(n)$.

Agora precisamos saber qual parte do eixo da qüefrência corresponde a $c_e(n)$ e qual parte corresponde a $c_\theta(n)$, pois, sabemos que eles estão separados. Intuitivamente, podemos concluir que a resposta de $\theta(n)$ estará situada em baixas qüefrências e a resposta de e(n) em altas qüefrências. Esta conclusão vem do fato que s(n) no domínio da freqüência é dado por uma envoltória (envelope) que limita pulsos bastante rápidos quando comparados com a envoltória. Se pudéssemos separar na freqüência as respostas relativas a $\theta(n)$ e a e(n), veríamos que a envoltória representaria a resposta de $\theta(n)$ e os pulsos seriam a representação do sinal de excitação e(n). Aplicando-se o logaritmo, estaremos somando as respostas de rápida e lenta variação. Ao fazermos a transformada inversa de Fourier, notaremos que o logaritmo da resposta em freqüência de $\theta(n)$ ocupará uma faixa de baixa qüefrência, pois, na freqüência ele tem uma lenta variação, e o logaritmo da resposta em freqüência de e(n) ocupará uma faixa de alta qüefrência (na forma de impulsos) devido a resposta em freqüência possuir rápida variação.

Com a conclusão acima, podemos simplesmente fazer um processo chamado de liftragem passa-baixas (PB), que seria o equivalente a uma filtragem passa-baixas só que no domínio da qüêfrencia. Com esta liftragem PB eliminaríamos de $c_s(n)$ a componente $c_e(n)$, ficando só com a resposta de $\theta(n)$ no domínio da qüefrência (cepstrum). A largura da janela de liftragem é tal que seja capaz de eliminar $c_e(n)$ de $c_s(n)$, e esta largura ideal seria onde ocorresse o primeiro pulso devido a excitação, que seria no instante n relativo ao próprio pitch do sinal. A periodicidade destes pulsos seria o próprio período de pitch do sinal.

3.3.2 - Liftros e medidas de distância

A literatura nos apresenta dois tipos de liftros usados na extração dos coeficientes cepstrais. Um seria simplesmente uma janela retangular e o outro teria uma forma onde daria maior ênfase aos coeficientes centrais da janela e de-enfatizando os coeficientes que se situam nos extremos da janela.

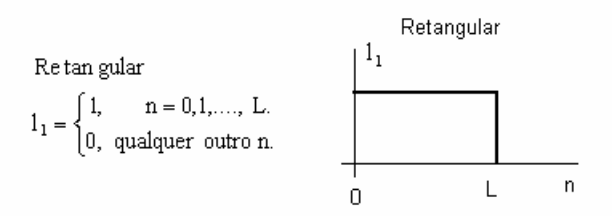


Figura 18: Liftro retangular.

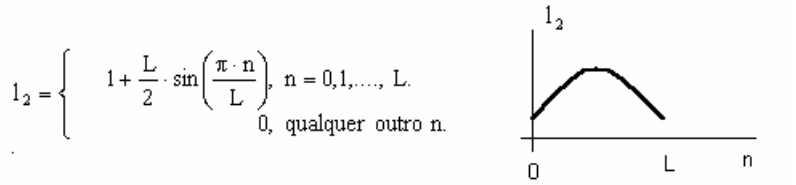


Figura 19: Liftro com pesagem maior para os coeficientes centrais.

Os dois tipos de distâncias usados estão relacionados com o tipo de liftro usado. Se usarmos o liftro l_2 (com pesagem), é aconselhável que usemos a distância Euclidiana, já se usarmos o liftro retangular é aconselhável o uso de uma distância cepstral com pesagem. Abaixo, formularemos estas distâncias.

Distância Euclidiana :

$$d_2[\underline{c}_{\hat{\theta}1}, \underline{c}_{\hat{\theta}2}] = \sqrt{[\underline{c}_{\hat{\theta}1} - \underline{c}_{\hat{\theta}2}]^T \cdot [\underline{c}_{\hat{\theta}1} - \underline{c}_{\hat{\theta}2}]}.$$

Distância cepstral com pesagem :

$$d_{2w}[\underline{c}_{\hat{\theta}1}, \underline{c}_{\hat{\theta}2}] = \sqrt{[\underline{c}_{\hat{\theta}1} - \underline{c}_{\hat{\theta}2}]^T \cdot \Lambda^{-1} \cdot [\underline{c}_{\hat{\theta}1} - \underline{c}_{\hat{\theta}2}]}$$

onde :

$$\underline{C}_{\hat{\theta}} = \underline{\Phi} \cdot \underline{\Lambda} \cdot \underline{\Phi}^T,$$

$\underline{C}_{\hat{\theta}}$ => Matriz de covariância de $\hat{\theta}$.

$\underline{\Phi}$ => Matriz ortonormal cujas colunas são os autovetores normalizados.

$\underline{\Lambda}$ => Matriz diagonal de autovalores (a diagonal são as variâncias dos coeficientes, individualmente).

Todos estes conceitos foram estabelecidos após um estudo feito por Tohkura(1987), onde ele mostrou que podemos obter melhores resultados utilizando estas técnicas acima apresentadas.

É importante levar em consideração uma outra medida que também pode ser muito útil em sistemas de reconhecimento. Esta medida se chama Delta cepstrum, e nada mais é que uma medida da variação entre parâmetros de segmentos adjacentes, deste carregamos informação da dinâmica comportamental dos parâmetros de segmento a segmento. Esta medida costuma ser incluída na representação de sinal. Ela é dada por: $\Delta c_a(n) \overset{def}{=} c_{a+\delta Q}(n) - c_{a-\delta Q}(n),$ onde a é o segmento em questão e δQ são

quantos segmentos antes e depois do segmento a vamos incluir no cálculo da variação Δ.

3.4 – Mel-Cesptrum

Abrimos uma nova seção para uma idéia que se situaria dentro da análise cepstral, porém, devido a sua importância e curiosidade resolvemos separá-la. Esta idéia se chama Mel-cepstrum e se baseia em mapearmos o sinal de fala não com a freqüência real e sim com a freqüência percebida pelo nosso aparelho auditivo.

Primeiramente explicaremos o que é mel. Mel é uma unidade de freqüência ou pitch percebido em um tom. Volkman e Stevens (1940) foram os pioneiros neste tipo de trabalho, e fizeram o seguinte: arbitraram que 1000 Hz é igual a 1000 mels, e realizaram testes com pessoas aumentando a freqüência real (Hz) até que as pessoas notassem um aumento na freqüência percebida de 2, 3, …,10 vezes e o mesmo fora feito diminuindo a freqüência. Eles descobriram que o mapeamento era linear abaixo de 1000 Hz e logarítmico acima de 1000 Hz.

Encontramos em algumas fórmulas que tentam relacionar F_{Hz} com F_{mel}. Abaixo mostraremos as formulações obtidas nas referências [6] e [4].

De acordo com a Voicebox [6], a relação é: $F_{mel} = \dfrac{\ln\left(1 + \dfrac{F_{Hz}}{700}\right) \cdot 1000}{\ln\left(1 + \dfrac{1000}{700}\right)}$.

De acordo com a referência [4], a relação é: $F_{mel} = 1000 \cdot \log_2 (1 + F_{Hz})$.

Plotando as duas fórmulas veremos que seus resultados são bastante parecidos, a diferença aumenta conforme aumenta a F_{Hz}. Para o nosso caso veremos que só nos interessará a relação até aproximadamente 4000 Hz, pois, é a metade da freqüência de amostragem que utilizamos (Fs = 8000 Hz).

Curiosamente as implementações dos coeficentes mel-cepstrais dos livros de Deller [1] e Rabiner [3] são diferentes e por conseguinte nos levam a resultados diferentes quando aplicadas a um sistema de reconhecimento de dígitos isolados. A seguir mostraremos as diferentes implementações.

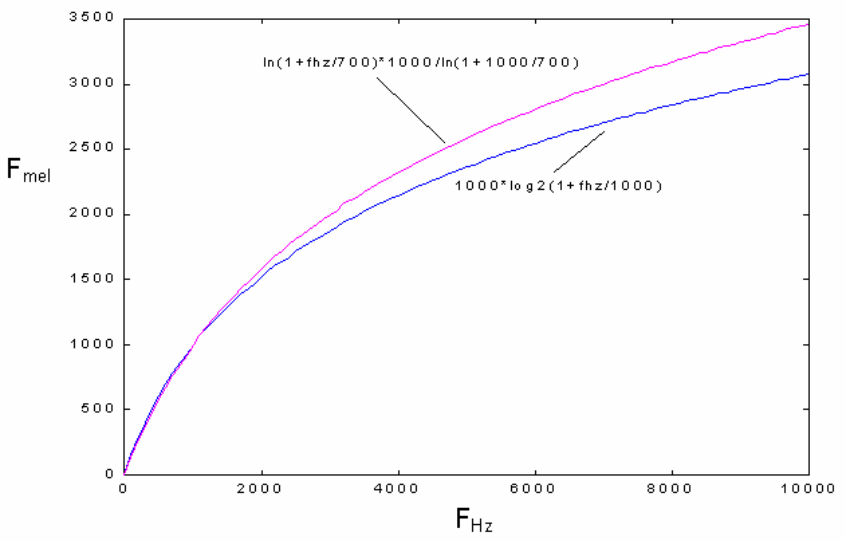


Figura 20: Relação entre F_{Hz} e F_{mel}.

Primeiramente existe a necessidade de se implementar um filtro, que é compostos por vários filtros chamados de banda critica (*"critical-band"* - cb) com largura de banda que aumenta logaritmicamente conforme o aumento da freqüência. A freqüência máxima do extremo superior do último filtro cb é no máximo a metade da freqüência de amostragem.

Abaixo mostraremos o gráfico do filtro cb. Na implementação podemos escolher a forma do filtro: triangular, blackman ou hanning, porém, na literatura só encontramos o tipo triangular. A idéia de se usar outros tipos de formas para os filtros foi vista na Voicebox [6].

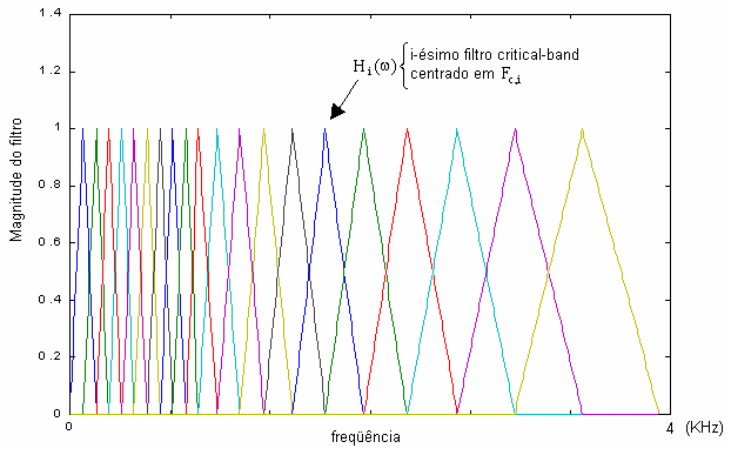


Figura 21: Filtros de banda critica.

O livro de Deller apresenta o diagrama adiante para efetuarmos o cálculo dos coeficientes mel-cepstrais. As freqüências centrais de cada filtro são dadas por $F_{c,i} = k \cdot \frac{Fs}{N'}$, onde N´ é o número de pontos usados no cálculo da DFT de $s(n) \cdot w(n-m)$ (sinal janelado), Fs é a freqüência de amostragem e k

representa os pontos no eixo horizontal de S(k,m) que é a DFT de $s(n) \cdot w(n-m)$.

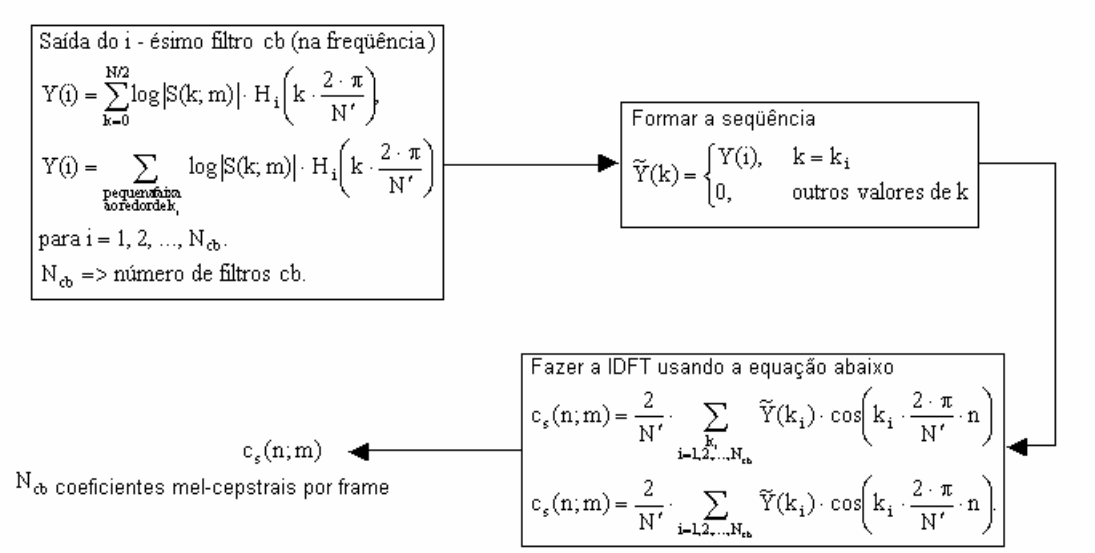


Figura 22: Diagrama esquemático do cálculo dos coeficientes mel-cepstrais pelo livro de Deller [1].

No livro de Rabiner [3], os coeficientes são simplesmente calculados através da fórmula: $\tilde{c}_n = \sum_{k=1}^{K} \left(\log(\tilde{S}_k)\right) \cdot \cos\left[n \cdot \left(k - \frac{1}{2}\right) \cdot \frac{\pi}{K}\right]$, $n = 1, 2, ..., N_{cb}$. É claro que estamos utilizando a mesma nomenclatura usada anteriormente, e fazemos este cálculo para cada segmento individualmente. $\tilde{S}_k$ é o espectro de potência do segmento do sinal s(n) já filtrado pelos filtros cb, onde k são os índices da DTFT da potência de s(n) e K é o maior índice a ser considerado. Os filtros são idênticos aos do cálculo anterior inclusive, nos dois livros, aconselham a utilização de filtros espaçados de 150 mels e com largura de banda de 300 mels. Este cálculo dos coeficientes mel-cepstrais não é claro quanto a filtragem do segmento de s(n). Basta utilizarmos um liftro PB com pesagem para pegarmos o número desejado de coeficientes por segmento (idêntico ao dos coef. cepstrais).

A representação de sinais não precisa utilizar somente os coeficientes extraídos por algum dos métodos acima explicados, podemos também usar energia e os deltas para complementar a representação do sinal. No trabalho proposto, utilizamos energia, delta-energia, delta-delta-energia, coeficientes, delta-coeficientes e delta-delta-coeficientes.

No algoritmo implementado em nosso trabalho podemos escolher se queremos os filtros espaçados como mencionado acima, ou escolher o número de filtros dentro da faixa que vai de 0 a Fs/2, ou ainda optarmos pela largura de banda dos filtros em mels, sabendo que eles serão espaçados da metade da largura de banda.

4 – DTW (*"Dynamic Time Warping"*)

DTW é uma técnica amplamente usada em reconhecimento de sinais de fala, sendo esta baseada na comparação com uma certa referência. Devido a necessidade de se ter uma referência para cada classe a ser reconhecida, esta técnica não é usada para tarefas complexas que envolvam grandes vocabulários. Em geral, utiliza-se para sistemas de reconhecimento de palavras isoladas e dependentes do orador.

O nome DTW vêm do fato de usarmos programação dinâmica como ferramenta para que o custo computacional diminua e de alinharmos o sinal de teste com o de referência tanto "esticando" como "comprimindo" no tempo.

O objetivo final do algoritmo DTW é comparar um sinal de referência com o sinal de teste que queremos reconhecer e gerar um resultado de similaridade entre estes sinais. No final este algoritmo gera uma medida de distância/similaridade total entre os dois sinais que pode ser interpretada como: quanto menor for esta distância mais parecidos serão os sinais.

4.1 – Programação dinâmica (DP)

Programação dinâmica é um método bastante utilizado em reconhecimento de sinais de fala, e trata-se de um conceito matemático para analisar processos que envolvam decisões ótimas.

Inicialmente, explicaremos de forma sucinta os princípios da DP. Observando o gráfico (fig.23) a seguir, notamos dois eixos j e i que podem representar qualquer coisa. No caso do nosso trabalho, eles representam os vetores de representação do sinal para os segmentos de 1 a I do sinal de teste e os mesmos vetores para os segmentos de 1 a J do sinal de referência. Isto foi só para termos uma idéia concreta. Os valores tanto para i quanto para j precisam ser maiores ou igual a zero.

O objetivo da DP é encontrar o caminho de "menor custo" ou "menor distância" através dos pontos mostrados no gráfico, que começa no nó (0,0) e termina no nó (I,J). O caminho é dado por: $(s,t),(i_1,j_1),(i_2,j_2),....,(u,v)$, onde (i_k,j_k) são nós intermediários e não possuem qualquer restrição. O caminho completo é dado se $(s,t)\equiv(0,0)$ e $(u,v)\equiv(I,J)$. Existem três tipos distâncias relacionadas ao caminho. As distâncias de transição (T), de nó (N) e de ambos (B – *"both"*).

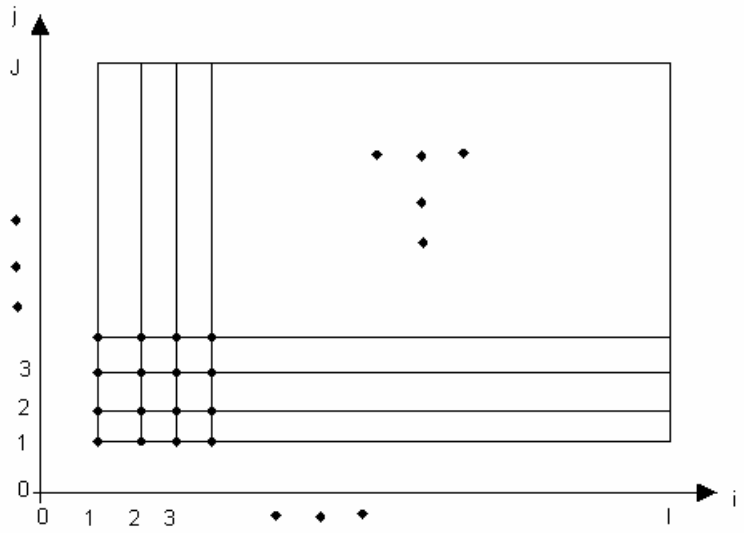


Figura 23: Gráfico para exemplificar Programação dinâmica.

Definindo os tipos de distâncias/custos:

- **Transição** (associado ao caminho percorrido do nó atual até o seu predecessor)
  $d_T[(i_k, j_k)|(i_{k-1}, j_{k-1})] \overset{def}{=}$ custo de transição
  Assumimos que :
  $d_T[(i, j)|(0,0)] = 0$ para todo (i, j); porém variações podem ocorrer.

- **Nó** (associado ao nó atual, a posição corrente)
  $d_N(i, j) \overset{def}{=}$ custo associado ao nó (i, j)
  Assumimos que :
  $d_N(0,0) = 0$, porém variações podem ocorrem.

- **Ambos** (*"Both"*) (como o próprio nome já diz, esta sofre influencia dos dois tipos anteriores)
  $d_B[(i_k, j_k)|(i_{k-1}, j_{k-1})] \overset{def}{=} d_T[(i_k, j_k)|(i_{k-1}, j_{k-1})] + d_N(i, j)$

Geralmente usamos o tipo B, pois, esta é a mais genérica das distâncias e associa os tipos T e N. Iremos denotar a distância d_B por d, sem o B subscrito para facilitar a nosso trabalho na hora de escrever. A distância associada ao caminho completo é dada por: $D = \sum_{k=1}^{K} d[(i_k, j_k)|(i_{k-1}, j_{k-1})]$, onde K é o número de transições do caminho e $(i_0, j_0) \equiv (0,0)$ e $(i_K, j_K) \equiv (I, J)$. O objetivo deste processo é encontrar um caminho que minimize D.

4.1.1 - Princípio de otimização de Bellman (BOP)

Este princípio tem um papel muito importante no algoritmo de programação dinâmica, tratando do conceito de melhor caminho que nada mais é que o caminho de custo mínimo. O princípio diz que: o melhor caminho para se ir do ponto (s,t) até o ponto (u,v) passando por (w,x) é dado pela concatenação dos melhores caminhos de (s,t) até (w,x) e de (w,x) até (u,v). Definindo matematicamente:

$$(s,t) \overset{(w,x)}{\vec{*}} (u,v) = (s,t) \vec{*} (w,x) \oplus (w,x) \vec{*} (u,v)$$

$\vec{*}$ - denota o melhor caminho.

$\oplus$ - denota concatenação.

Desta forma podemos definir:

$$D_{min}(i,j) \overset{def}{=} \text{distância de } (0,0) \text{ até } (i,j) \text{ sobre o melhor caminho}$$

$$= \text{distância associada } (0,0) \vec{*} (i,j).$$

e

$$D_{min}[(i_k, j_k) | (i_{k-1}, j_{k-1})] \overset{def}{=} \text{distância de}(0,0) \text{ até } (i_k, j_k) \text{sobre o melhor caminho parcial passando por } (i_{k-1}, j_{k-1})$$

$$= \text{distância associada a } (0,0) \overset{(i_{k-1}, j_{k-1})}{\vec{*}} (i_k, j_k)$$

Como conseqüência direta temos que o custo mínimo total para chegar até o nó (i_k, j_k) passando pelo nó (i_{k-1}, j_{k-1}) é igual a distância mínima total para se chegar ao nó (i_{k-1}, j_{k-1}) mais a distância $d[(i_k, j_k) | (i_{k-1}, j_{k-1})]$. Matematicamente temos:

$$D_{min}[(i_k, j_k) | (i_{k-1}, j_{k-1})] = D_{min}(i_{k-1}, j_{k-1}) + d[(i_k, j_k) | (i_{k-1}, j_{k-1})].$$

Temos que o melhor caminho para se chegar ao nó (i_k, j_k) é a minimização de $D_{min}[(i_k, j_k) | (i_{k-1}, j_{k-1})]$ tirada sobre todos o possíveis predecessores de (i_k, j_k), ou seja,

$$D_{min}(i_k, j_k) = \min_{(i_{k-1}, j_{k-1})} \{D_{min}(i_{k-1}, j_{k-1}) + d[(i_k, j_k) | (i_{k-1}, j_{k-1})]\}.$$

Arbitramos $D_{min}(i_k, j_k) \overset{def}{=} 0$.

Uma vez que o caminho ótimo é encontrado, nós simplesmente recorremos a memória dos nós predecessores do caminho ótimo parcial para podermos rastrear os nós por onde passamos para chegar neste caminho ótimo. Definimos uma função de índice do nó predecessor.

$$\Psi(i_k, j_k) \overset{def}{=} \text{ índice do nó predecessor ao nó } (i_k, j_k) \text{ no } (0,0) \vec{*} (i_k, j_k).$$

$$\Psi(i_k, j_k) = (i_{k-1}, j_{k-1}).$$

Algumas restrições quanto a forma de caminhar devem ser obedecidas pois, não faz sentido caminhar no sentido decrescente do eixo i ou do eixo j. Já que o objetivo é saindo do nó (0,0) chegar no nó (I,J).

4.2 - Aplicando ao problema de reconhecimento de palavras isoladas

Como já mencionamos anteriormente, os ponto no eixo i representam vetores de características do sinal de teste e os pontos do eixo j tem o mesmo significado, só que é relativo a referência. No nosso trabalho, estes vetores serão a representação do sinal para cada segmento do sinal de teste e do sinal de referência. A notação será dada por:

$$\text{características do sinal de teste} : \underline{t}(1), \underline{t}(2), ..., \underline{t}(i), ..., \underline{t}(I)$$
$$\text{características do sinal de referência} : \underline{r}(1), \underline{r}(2), ..., \underline{r}(j), ..., \underline{r}(J)$$

Todos os conceitos aplicados em DP devem ser transportados para esta seção com certas ressalvas. Como por exemplo, d_N continua sendo maior que zero porém representa a distância(medida similaridade) entre um vetor $\underline{t}$ e um vetor $\underline{r}$.

$$0 \le d_N(i_k, j_k) = \text{custo relativo a comparação de similarideda entre } \underline{t}(i_k) \text{ e } \underline{r}(j_k).$$

Se $\underline{t}$ e $\underline{r}$ forem vetores de predição linear usamos $d_N(i_k, j_k) = d_I\left[\underline{t}(i_k), \underline{r}(j_k)\right]$. Já se usarmos coeficientes cepstrais com pesagem teremos $d_N(i_k, j_k) = d_2\left[\underline{t}(i_k), \underline{r}(j_k)\right] = \left\|\underline{t}(i_k) - \underline{r}(j_k)\right\|^2$.
Graficamente teremos:

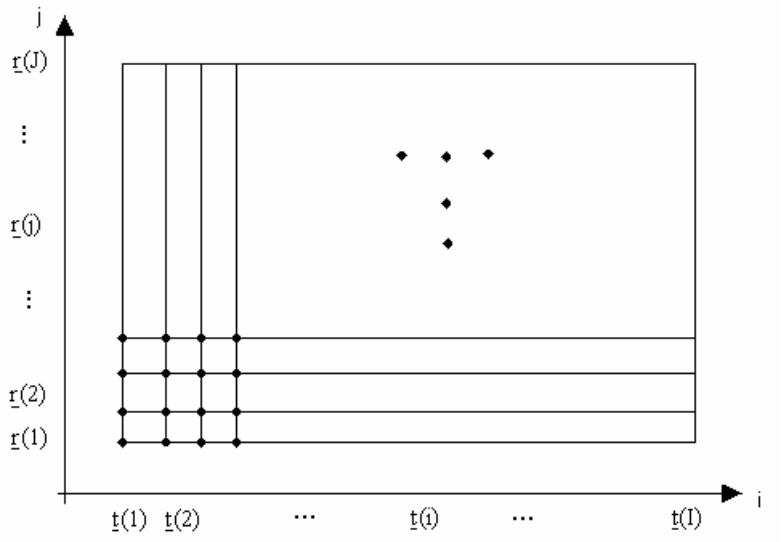


Figura 24: Características dos vetores de teste e referência associadas as coordenadas i e j.

O objetivo continua o mesmo do apresentado para a programação dinâmica, ou seja, encontrar o melhor caminho saindo do ponto $(\underline{t}(1), \underline{r}(1))$ e chegando no ponto $(\underline{t}(I), \underline{r}(J))$. Esta restrição nos deixa sem muita "mobilidade", por exemplo: se o recorte do sinal não foi "perfeito" estaremos incorrendo num erro de comparar segmentos iniciais e finais do teste e da referência que não correspondem propriamente ao início, nem ao fim da palavra a ser reconhecida. Uma forma de resolver isto, é "relaxar" esta restrição dando a liberdade de se começar em qualquer nó da área limitada pelos pontos $(\underline{t}(1), \underline{r}(1))$, $(\underline{t}(1), \underline{r}(1 + A))$, $(\underline{t}(1 + B), \underline{r}(1))$ e $(\underline{t}(1 + B), \underline{r}(1 + A))$, e terminar em qualquer nó da área limitada pelos pontos $(\underline{t}(I - D), \underline{r}(J - C))$, $(\underline{t}(I), \underline{r}(J - C))$, $(\underline{t}(I - D), \underline{r}(J))$ e $(\underline{t}(I), \underline{r}(J))$.

A monotonicidade é uma restrição que deve ser obedecida pois, estabelece os caminhos ilegais dentro do gráfico de pontos (i,j). Esta restrição diz que $i_{k-1} \leq i_k$ e $j_{k-1} \leq j_k$. Caso não estabelecêssemos esta restrição, poderíamos ter "compressões" ou "expansões" (warpings) impróprias no caminho encontrado.

As restrições globais de caminho limitam a região total de procura do melhor caminho considerando um limite máximo tanto para compressão quanto para expansão. Estas restrições facilitam bastante o trabalho computacional, e eliminam pontos que são muito improváveis de pertencer ao melhor caminho, como por exemplo, pontos do extremo superior esquerdo (1,J) e do extremo inferior direito (I,1) do gráfico. A seguir, mostraremos graficamente dois tipos diferentes de restrições globais.

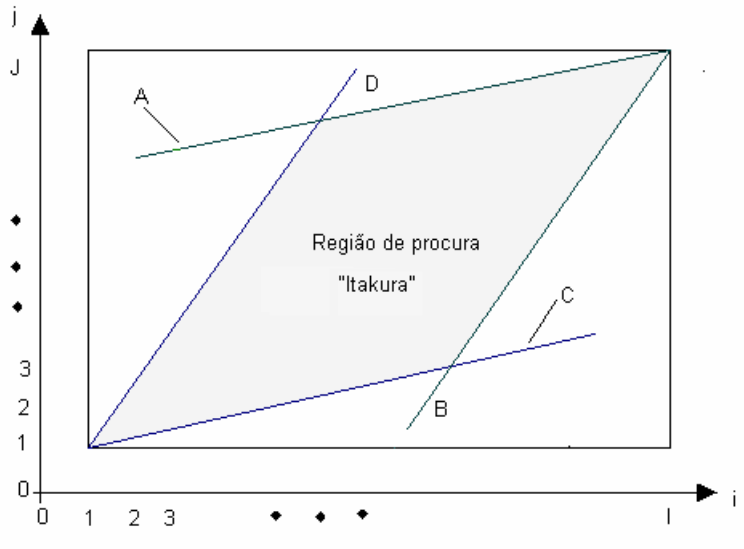


Figura 25: Restrição de Itakura para caminho global para um fator 2 como máximo de expansão e compressão.

As retas A, B, C e D são dadas por:

$$A: j = \frac{1}{2} \cdot i - (\frac{1}{2} \cdot I + J),$$

$$B: j = 2 \cdot i - (2 \cdot I - J),$$

$$C: j = \frac{1}{2} \cdot i + \frac{1}{2},$$

$$D: j = 2 \cdot i - 1.$$

A restrição de caminho global de Itakura limita uma região de procura para no máximo compararmos um sinal duas vezes maior que o outro e no mínimo a metade do outro. Existe uma restrição um pouco mais simples que limita uma região de procura através de uma parâmetro $W \geq |I - J|$, ou seja, o parâmetro W depende da diferença entre o número de segmentos do sinal de teste e do sinal de referência. Este tipo de restrição foi usada no algoritmo implementado neste trabalho pois, se usássemos a limitação de Itakura não poderíamos comparar sinais que fossem menores que a metade ou maiores que o dobro um do outro, e isto faria com que precisássemos gravar novos sinais quando este problema fosse detectado. E como mencionado, o Matlab não nos disponibiliza recursos para fazermos gravações, estaríamos complicando o problema de reconhecimento.

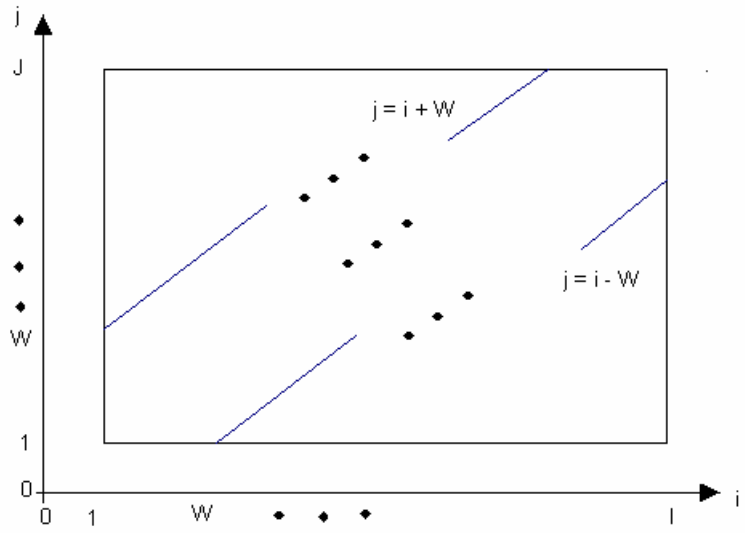


Figura 26: O tipo de restrição mais simples para limitar o caminho global.

As restrições locais de caminho limitam as possíveis transições para se passar de um nó para o subseqüente e nos informa o peso que cada transição pode ter. Elas especificam claramente o custo de transição d_T do caminho

percorrido. Não mostraremos todos os tipos de restrições encontradas na literatura mas, somente as que utilizamos em nosso trabalho.

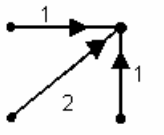


Figura 27: O tipo de limitação local mais simples. Os valores nos vetores da figura indicam a pesagem de cada transição.

Assumindo que todos os três vetores estejam convergindo para o ponto (i_k, j_k). Então a distância D total referente a este caminho parcial é dada por:

$$\min\begin{Bmatrix} D(i_{k-1}, j_k) + d(i_k, j_k) \\ D(i_{k-1}, j_{k-1}) + 2 \cdot d(i_k, j_k) \\ D(i_k, j_{k-1}) + d(i_k, j_k) \end{Bmatrix}$$

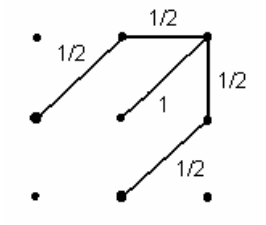


Figura 28: Outro tipo de restrição local mais complexa que a anterior, possibilitando transições entre nós mais distantes.

Neste caso D é dado por:

$$\min\begin{Bmatrix} D(i_{k-2}, j_{k-1}) + \frac{1}{2} \cdot [d(i_{k-1}, j_k) + d(i_k, j_k)] \\ D(i_{k-1}, j_{k-1}) + 1 \cdot d(i_k, j_k) \\ D(i_{k-1}, j_{k-2}) + \frac{1}{2} \cdot [d(i_k, j_{k-1}) + d(i_k, j_k)] \end{Bmatrix}$$

4.3 - Explicando o funcionamento do algoritmo DTW implementado

Tendo em mãos os coeficientes que representam cada segmento do sinal de referência e de teste, então geramos o gráfico explicado anteriormente onde colocamos o sinal de teste no eixo horizontal e o de referência no eixo vertical.

O próximo passo é limitar a região de procura global, através do tipo de restrição mais simples vista na anteriormente, devemos ainda utilizar algum tipo de restrição local de procura.

Para todos os pontos começando em (1,1) calculamos D, considerando o tipo de restrição local utilizada conforme é mostrado acima (figs. 26 e 27).

Assumimos que D para os pontos situados na coluna 1, ou seja, de (1,1) até (1,W) sejam iguais a zero. Então fazemos o mesmo para todos os nós da coluna 2 que estejam dentro da região de procura global, tudo isso obedecendo a restrição de procura local. Isto é feito sucessivamente para todos os nós que estão dentro da região de procura global até chegarmos na coluna I.

Assumimos que o ponto (I,J) é ponto final do melhor caminho e fazemos o caminho contrário em direção a origem. Quando chegarmos num nó onde a coluna ou a linha é 1 (ou ambos) paramos; pois, já teremos encontrado o melhor caminho. O número de nós por onde passamos no melhor caminho é dado por M. A distância final é dada pela divisão entre D que é o somatório das distâncias locais e de transição entre os nós do melhor caminho e M. Podemos considerar este procedimento como sendo uma espécie de normalização.

$$D_{final} = D(I,J) / M.$$

Como pudemos notar existe um certo relaxamento quanto ao nó inicial do caminho de menor custo total, porém, não temos este mesmo tipo de relaxamento para o nó final. Sempre atrelamos o nó final a (I,J). No final temos um valor D_{final} e quanto menor for este valor indicará que mais parecido são os sinais comparados. D_{final} é um valor de similaridade entre dois sinais de tamanhos diferentes. Este algoritmo foi baseado no livro e nas notas do sistema de reconhecimento implementado pelo grupo de pesquisa em processamento de sinais de fala da Universidade do Porto – Portugal, principalmente nas notas e algoritmos implementado pelo Prof. Carlos Espain [7].

4.4 – Algoritmo de treinamento e reconhecimento

Estes dois algoritmos são baseados no algoritmo DTW pois o treinamento e o reconhecimento nada mais são que comparações entre sinais, e o que gerar o resultado de D_{final} mais satisfatório em comparação aos outros (dependendo do tipo de aplicação) será o sinal desejado.

4.4.1 - Algoritmo de treinamento

Sabemos que cada classe tem um certo número de elementos, considerando o nosso trabalho, as classes são os dígitos que queremos reconhecer {0, 1, …, 9} e o número de elementos de cada classe é 5, ou seja, cinco sinais de treinamento para cada classe.

O algoritmo de treinamento tem por intuito encontrar o elemento que melhor representa a classe. Este será a nossa referência ou padrão. Intuitivamente consideramos o elemento que melhor representa a classe como sendo o elemento mais central da classe.

Para encontrarmos este sinal mais representativo, devemos calcular a distância D_{final} de um elemento para todos os outros e fazermos o somatório de D_{final}; este somatório é feito para todos os elementos da classe. Depois, pegamos o elemento que tenha o menor somatório como sendo o mais representativo. Representamos por $t_k(n)$ o elemento k de uma certa classe, sendo que: $1 \le k \le 5$ e t_R como sendo o sinal mais representativo da classe. Matematicamente temos:

$$S(j) = \sum_{k=1}^{5} D_{final}(t_1(n), t_k(n))$$

$$t_R(n) \text{ será tal que } S(R) < S(j), \forall j \mid j \neq R \text{ e } 1 \le j \le 5.$$

Outros valores que utilizaremos no algoritmo de reconhecimento serão a maior distância entre dois elementos da classe(dmax1) e a maior distância entre o elemento mais representativo e um outro elemento da classe(dmax). Graficamente teremos uma idéia melhor do que isto significa:

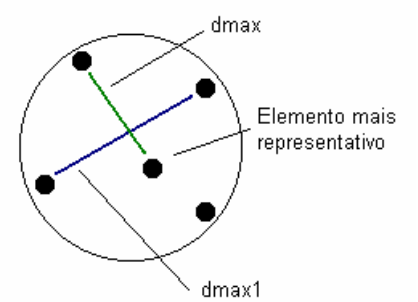


Figura 29: Elementos e valores calculados no algoritmo de treinamento.

No final teremos um sinal padrão ou de referência para cada classe com seus respectivos dmax e dmax1.

4.4.2 - Algoritmo de reconhecimento

Este algoritmo pega um sinal de teste e compara com o padrão de cada classe (centróide) através do algoritmo DTW, e o que resultar na menor distância D_{final} será reconhecido como pertencente a classe do padrão que gerou este D_{final}. Chamaremos $p_k(n)$ o sinal padrão (centróide) da classe k e g(n) o sinal de teste.

$$\min\{D_{final}(g(n), p_k(n)) \mid 1 \le k \le (\text{número de classes})\}$$

$$k \text{ relativo ao } D_{final} \text{ mínimo será a classe reconhecida.}$$

Também podemos recorrer aos elementos da classe, ou seja, quando fazemos o reconhecimento pegamos os dois menores D_{final} (as duas classes mais prováveis usando o centróide) e comparamos g(n) com todos os elementos destas duas classes mais prováveis. Fazemos o somatório das D_{final}'s para todos os elementos da classes e a que gerar menor somatório será

a classe reconhecida. Este procedimento também é executado para as cinco classes mais prováveis obtidas na comparação com o centróide.

Os valores de dmax e dmax1 são utilizados com o intuito de se diminuir o percentual de erros de reconhecimento e pelo menos “manter” o percentual de acertos. Podemos fazer um teste para nos poupar trabalho computacional, primeiro executamos o reconhecimento propriamente dito (comparação com os centróides de cada classe), depois pegamos o maior dmax ou o maior dmax1 de um certo conjunto de padrões (todas as classes) e comparamos com o D_{final} obtido pelo centróide da classe reconhecida com g(n) (sinal de teste). Se $A \cdot (\text{d max ou dmax1}) < D_{final}(\text{centróide da classe reconhecida}, g(n))$, então dizemos que este sinal g(n) esta fora da faixa de reconhecimento e pediríamos para entrar com um novo sinal de teste. O termo A mencionado a anteriormente, nada mais é que um fator de escala utilizado neste teste. O reconhecimento pode ser feito sem este teste, de modo que, sempre teremos um resultado para o reconhecimento, pois, em muitas aplicações é imprescindível que tenhamos alguma estimativa do resultado de reconhecimento.

5 - Experimentos

Nesta seção mostraremos alguns resultados e faremos uma análise deles. Como sabemos, existe um grande número de parâmetros que podem ser escolhidos no decorrer do reconhecimento, como por exemplo, o tipo de janela usada na segmentação, o tipo de restrição global e local usada no algoritmo DTW, qual o fator multiplicativo do teste para ver se o sinal de teste está dentro da faixa de reconhecimento e vários outros dependendo dos tipos de parâmetros usados para representar o sinal.

Nesta etapa inicial começaremos com tamanho de segmento 20 ms(tframe=20 ms), usaremos para representar cada segmento um valor de energia + um de delta-energia + um de delta-delta-energia + doze de coeficientes cepstrais + doze de delta-cepstrais e doze de delta-delta-cepstrais, dando um total de 39 parâmetros(39cc). A janela utilizada na segmentação é retangular sem superposição. A restrição local usada no algoritmo DTW é chamada de dtw11 que é a mais simples apresentada neste trabalho onde o nó antecessor só poderá ser um do três nós adjacentes ao nó em questão (ver fig. 27).
Chamaremos esta restrição de dtw11 e a outra de dtw22_1 (ver fig. 28).

A restrição global será dada por W caso $|I - J| > 2$, então $W = |I - J|$, caso contrário W será igual a 2. Nestes primeiros resultados estaremos testando um possível limiar para $(A \cdot d\max)$ ou $(A \cdot d\max 1)$ de modo que satisfaçam a condição proposta para este teste, que é diminuir o percentual de erro de reconhecimento e pelo menos manter o percentual de acertos. Outro fator muito importante a ser considerado é que estamos eliminando os sinais nos quais o recorte foi problemático. Isto significa que estaremos analisando somente a eficiência do sistema de reconhecimento e não do processamento total do sinal. Apesar do Matlab ser uma ferramenta fabulosa, ele não facilita o nosso trabalho caso desejemos desprezar alguns sinais mal recortados **(ver seção 2.4 – Tabela 7)**. Para solucionar este problema nós eliminaremos os sinais mal recortados e adicionaremos sinais bem recortados em seus lugares. Consequentemente, nós teremos um universo de treinamento com 50 sinais e com 100% de acerto na detecção de endpoint. Após obtermos os melhores resultados para os parâmetros a serem escolhidos, iremos analisar a eficiência do sistema total.

Os testes foram feitos utilizando-se os 50 sinais (5 elementos de cada classe) de treinamento mais 50 sinais de teste diferentes dos sinais de treinamento, sendo que todos estes sinais são de um só orador.

Experimento 1: Teste de limiar (A*dmax ou A*dmax1)

Usando:
- tframe=20 ms.
- janela (segmentação)=retangular.
- Sem superposição (segmentação)
- dtw11.
- 39cc(coef. cepstrais) por segmento=1e+1de+1dde+12cc+12dcc+12ddcc.

Como podemos notar os gráficos estão fornecendo os percentuais totais, ou seja, o percentual de acerto e de erro considerando os 100 sinais, 50 de treinamento e 50 de teste.

As tabelas são para **1 centróide**, **2 classes menores** significa que esta recorrendo a todos os elementos das duas classes mais prováveis e **5 classes menores** significa que estamos recorrendo a todos os elementos das 5 classes mais prováveis.

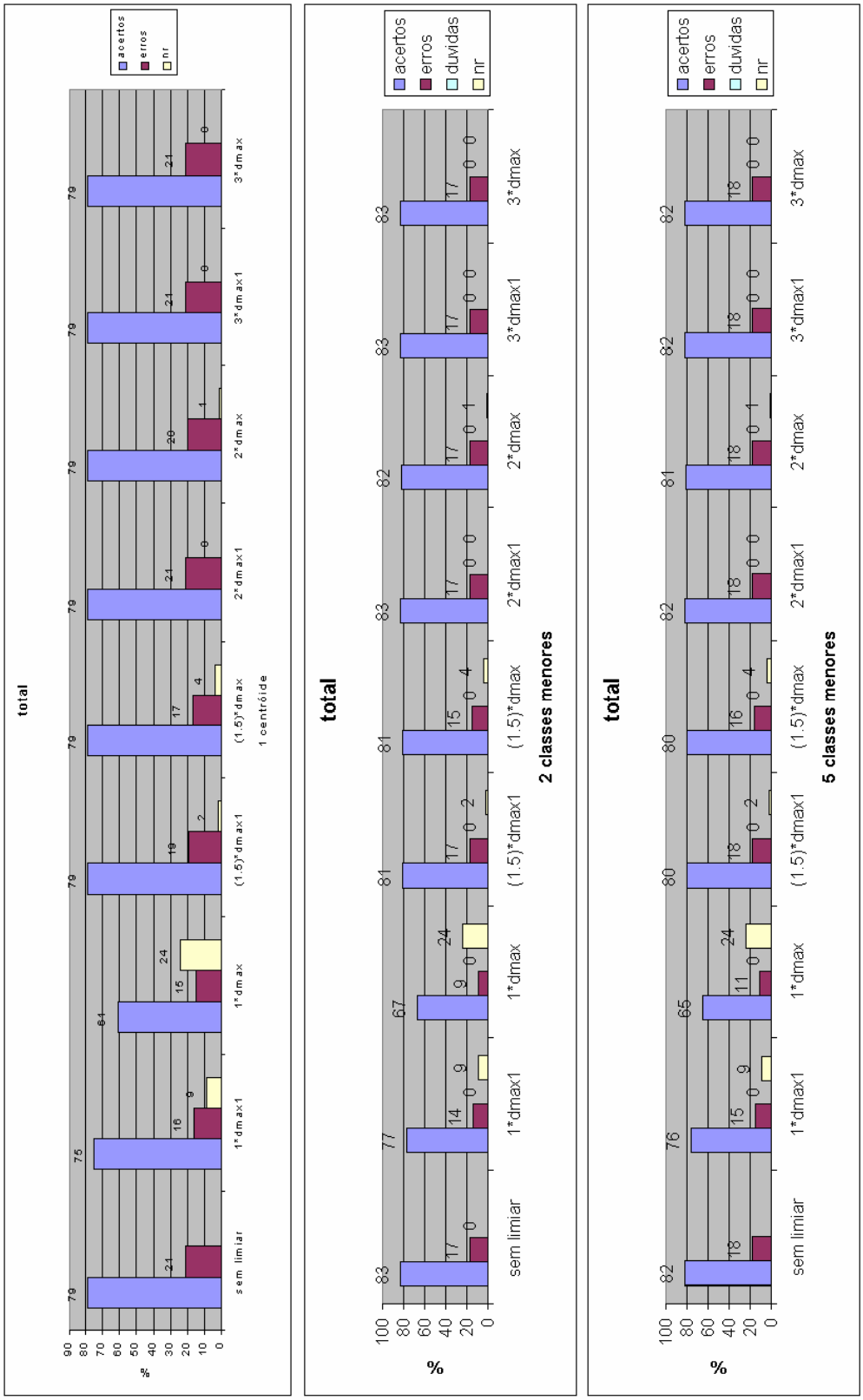
A palavra **total** significa que estamos analisando os 100 sinais, quando estiver escrito **treinamento** ou **universo de treinamento** significa que estamos analisando somente os 50 sinais de treinamento e quando for **universo de teste** ou somente **teste** estaremos analisando os 50 sinais de teste.

As palavras **acerto**, **erro** e **nr** se referem respectivamente aos percentuais de acerto, de erro e de não-reconhecidos, que na verdade são os sinais que se encontram fora da faixa de reconhecimento. As **dúvidas** ocorrem somente nos casos onde fazemos o reconhecimento com 2 classes menores e com 5 classes menores, pois, estas são relacionadas ao resultado do somatório das distâncias D_{final} entre cada elemento da classe e g(n) ser mínimo para duas classes diferentes. Neste caso teríamos duas classes igualmente prováveis. Vale a pena mencionar que este resultado(**dúvida**) é muito difícil de ocorrer.

Analisando a tabela notamos que: quando os valores de A são altos independentemente, se utilizamos dmax ou dmax1 os resultados são bem parecidos com os resultados quando não utilizamos limiar algum (todo sinal de teste gera um resultado de reconhecimento então para este caso nr=0). O melhor resultado foi obtido para o valor de limiar **(1,5)*dmax**, na verdade este foi o único que atendeu a consideração de manter o percentual de acertos e diminuir o percentual de erros, quando não, pelo menos diminui igualmente os valores dos percentuais de acertos e erros, não comprometendo o resultado final se comparado com o resultado obtido **sem limiar**. Este resultado, utilizando **(1,5)*dmax**, não foi muito melhor do que o resultado **sem limiar**.

Caso precisássemos utilizar algum limiar, deveríamos utilizar o valor acima mencionado como o de melhor resultado, porém, se for indiferente o uso de limiar, **parece** muito **melhor não estipularmos limiar algum**. O resultado

final não irá mudar muito e sempre teremos alguma estimativa sobre o reconhecimento do sinal de teste.



Experimento 2: Teste DTW (testando o tipo de restrição local)

Usando:

- tframe=20 ms.
- janela (segmentação)=retangular.
- Sem superposição (segmentação)
- 39cc(coef. cepstrais) por segmento=1e+1de+1dde+12cc+12dcc+12ddcc.
- Sem limiar.

Neste teste estamos analisando os resultados referentes às restrições locais de caminho, ou seja, estaremos limitando a distância de transição em alguns pontos. No caso de **dtw11**, estaremos limitando as transições entre os nós $(i-1, j)$, $(i, j-1)$ e $(i-1, j-1)$ sendo o nó subseqüente (i, j). No **dtw22_3** estaremos limitando as transições entre os nós $(i-2, j-1)$, $(i-1, j-2)$ e $(i-1, j-1)$ sendo o nó subseqüente (i, j) e os nós intermediários, $(i-1, j)$ e $(i, j-1)$, também precisam ser válidos pois entram no cálculo da distância. Este procedimento foi explicado com mais detalhes na seção do algoritmo DTW. O **dtw22_4** é uma variação do dtw22_3 pois, ao invés de considerar três transições, ele considera 5 devido a inclusão dos nós intermediários no cálculo. Esta variação não se encontra na literatura porém me pareceu importante fazê-la com o intuito de observar a sua influência no resultado.

Neste caso a distância B é dada por:

$$\min\begin{Bmatrix} D(i_{k-1}, j_k) + \frac{1}{2}\cdot[d(i_k, j_k)] \\ D(i_{k-2}, j_{k-1}) + \frac{1}{2}\cdot[d(i_{k-1}, j_k) + d(i_k, j_k)] \\ D(i_{k-1}, j_{k-1}) + 1\cdot d(i_k, j_k) \\ D(i_{k-1}, j_{k-2}) + \frac{1}{2}\cdot[d(i_k, j_{k-1}) + d(i_k, j_k)] \\ D(i_k, j_k) + \frac{1}{2}\cdot[d(i_k, j_k)] \end{Bmatrix}$$

Observando as tabelas a seguir, notamos que os resultados considerando dtw22_3 e dtw22_4 são melhores que o resultado referente a dtw11. Isto já era de se esperar pois dtw22_3 e dtw22_4 dão uma maior liberdade de procura do melhor caminho, ou seja, temos uma área maior para encontrar a presença do caminho de menor custo. Apesar de gerar melhores resultados há um certo custo que se deve pagar por isto, que é o aumento da complexidade computacional, incorrendo num algoritmo mais demorado. No teste de reconhecimento de 100 sinais, o algoritmo dtw11 é aproximadamente 10 min mais rápido que os algoritmos dtw22_3 e dtw22_4. Observando somente os resultados dos algoritmos dtw22_3 e dtw22_4, notamos que suas diferenças são “insignificantes”, sendo praticamente irrelevante levantar a questão sobre qual deles gera melhor resultado em menor tempo (observar a tabela de resultados totais e seus respectivos tempos de execução).

É uma **relação custo/benefício**, se quisermos melhorar o resultado alterando a restrição de caminho local, teremos que pagar um certo preço por isto, em forma de tempo de execução. Dependendo da aplicação para a qual estamos usando, este custo pode ser irrelevante, mas para aplicações que necessitam rapidez no processamento, este custo pode ser um preço muito caro a se pagar.

		Dtw11		Dtw22_3	
		Universo de treinamento	Universo de Teste	Universo de treinamento	Universo de Teste
1 centróide	Acertos:	92%	66%	98%	64%
	Erros:	8%	34%	2%	36%
	-	-	-	-	-
2 classes menores	Acertos:	96%	70%	100%	76%
	Erros:	4%	30%	0%	24%
	Dúvidas:	0%	0%	0%	0%
5 classes menores	Acertos:	96%	68%	100%	78%
	Erros:	4%	32%	0%	22%
	Dúvidas:	0%	0%	0%	0%

		Dtw22_4	
		Universo de treinamento	Universo de Teste
1 centróide	Acertos:	94%	68%
	Erros:	6%	32%
	-	-	-
2 classes menores	Acertos:	100%	78%
	Erros:	0%	22%
	Dúvidas:	0%	0%
5 classes menores	Acertos:	100%	80%
	Erros:	0%	20%
	Dúvidas:	0%	0%

		Dtw11	Dtw22_3	Dtw22_4
		Universo total	Universo total	Universo total
1 centróide	Acertos:	78%	81%	81%
	Erros:	22%	19%	19%
	-	-	-	-
2 classes menores	Acertos:	83%	88%	89%
	Erros:	17%	12%	11%
	Dúvidas:	0%	0%	0%
5 classes menores	Acertos:	82%	89%	90%
	Erros:	18%	11%	10%
	Dúvidas:	0%	0%	0%
	tempo	**30min**	**41min**	**42min**

Tabelas referentes ao **Experimento 2**.

Experimento 3: Teste de tamanho de segmento

Usando:

- janela (segmentação)=retangular.
- Sem superposição (segmentação).
- dtw11.
- 39cc(coef. cepstrais) por segmento=1e+1de+1dde+12cc+12dcc+12ddcc.
- Sem limiar.

Neste teste estamos analisando os resultados referente a diferentes tamanhos de janela de segmentação. Segmentos menores obviamente geram um maior números de segmentos para um mesmo sinal quando segmentado por segmentos maiores. Evidentemente quanto maior o número de segmentos por sinal maior será o custo computacional e por conseguinte maior será o tempo gasto na execução.

Estes testes têm por intuito estipular um tamanho de segmento que seja ótimo para o algoritmo de reconhecimento. Como sabemos, é necessário que segmentemos o sinal em segmentos relativamente pequenos onde tais segmentos do sinal de fala possam ser considerados aproximadamente estacionários.

Observando os resultados das tabelas subseqüentes, notamos que os melhores resultados foram obtidos com segmentos de 15 ms porém estes resultados não são significativamente melhores em comparação aos resultados usando os valores de 20 ms e 10 ms para o tamanho dos segmentos. Podemos observar na tabela de resultados totais que o tempo de execução aumenta muito conforme se diminui o tamanho do segmento.

De forma geral, não me parece vantajoso utilizar segmentos de tamanho menor que 20 ms pois, o tempo de execução de um total de 100 sinais é menor em 10 min que o de 15 ms e 45 min menor que o de 10 ms e seus resultados não são tão ruins se comparados com os resultados obtidos utilizando os outros dois tamanhos de segmentos. Porém o fato de ser vantajoso ou não, depende da aplicação na qual estamos usando o sistema.

		10 ms		15 ms	
		Universo de treinamento	Universo de Teste	Universo de treinamento	Universo de Teste
1 centróide	Acertos:	86%	64%	94%	72%
	Erros:	14%	36%	6%	28%
	-	-	-	-	-
2 classes menores	Acertos:	92%	72%	96%	72%
	Erros:	8%	28%	4%	28%
	Dúvidas:	0%	0%	0%	0%
5 classes menores	Acertos:	100%	78%	96%	72%
	Erros:	0%	22%	4%	28%
	Dúvidas:	0%	0%	0%	0%

		20 ms	
		Universo de Treinamento	Universo de Teste
1 centróide	Acertos:	92%	66%
	Erros:	8%	34%
	-	-	-
2 classes menores	Acertos:	96%	70%
	Erros:	4%	30%
	Dúvidas:	0%	0%
5 classes menores	Acertos:	96%	68%
	Erros:	4%	32%
	Dúvidas:	0%	0%

		10 ms	15 ms	20 ms
		Universo total	Universo Total	Universo total
1 centróide	Acertos:	75%	83%	78%
	Erros:	25%	17%	22%
	-	-	-	-
2 classes menores	Acertos:	82%	84%	83%
	Erros:	18%	16%	17%
	Dúvidas:	0%	0%	0%
5 classes menores	Acertos:	89%	84%	82%
	Erros:	11%	16%	18%
	Dúvidas:	0%	0%	0%
	Tempo	**75min**	**41min**	**30min**

Tabelas referentes ao **Experimento 3**.

Experimento 4: Teste do tipo de coeficiente

Usando:
- tframe=20 ms.
- janela (segmentação)=retangular.
- Sem superposição (segmentação).
- dtw11.
- 39c(coef.) por segmento=1e+1de+1dde+12c+12dc+12ddc.
- filtros cb triangulares (tamanho variável)
- Sem limiar.

Neste teste estamos analisando os resultados referente a diferentes tipos de coeficientes utilizados para representar o sinal. No caso, fazemos somente a comparação entre coeficientes cepstrais e mel-cepstrais pois, a maioria dos trabalhos publicados na área utilizam estes tipos de coeficientes uma vez que geram resultados melhores que os coeficientes LP.

Nas primeiras tabelas temos: uma com resultado devido a coeficientes cepstrais **(base de comparativa – a mesma tabela do teste de tamanho de segmento usando 20ms)**, e outras três com resultados referentes a coeficientes mel-cepstrais implementados de acordo com a fórmula dada no livro de Deller [1] **(variação 0)** e as **variações 1 e 2** são relativas a implementação da **variação 0** (apresentada na **figura 22** deste trabalho), só que com pequenas alterações. Estas alterações são respectivamente a retirada do fator multiplicativo (2/N') da equação $c_s(n;m) = \frac{2}{N'} \cdot \sum_{i=1,2,...,N_{cb}} \tilde{Y}(k_i) \cdot \cos\left(k_i \cdot \frac{2 \cdot \pi}{N'} \cdot n \right)$ e a não utilização desta e equação para o cálculo de $c_s(n;m)$, usamos diretamente a DCT do Matlab aplicada a $\tilde{Y}(k)$ pois, na referência [7] utilizam a DCT para a obtenção dos coeficientes mel-cepstrais. Esta variações foram tentativas de se obter melhores resultados uma vez que os coeficientes cepstrais ainda estão gerando resultados melhores que os coeficientes mel-cepstrais até então implementados. Não conseguimos obter resultados melhores que o obtido com os coeficientes cepstrais. Nas **variações 0, 1 e 2 usamos 25 filtros cb triangulares**.

As **variações de 3 a 6** são outras tentativas de se conseguir melhores resultados utilizando coeficientes mel-cepstrais. A implementação é feita calculando-se $\tilde{Y}(k)$ pela fig. 21 a única diferença é que utilizamos o espectro de potência do sinal s(n) e não mais o espectro puro e simples do sinal s(n), e depois aplicamos este resultado na equação modificada da implementação do Rabiner [3] dada por: $\tilde{c}_n = \sum_{k=1}^{K} \left(\tilde{Y}(k) \right) \cdot \cos\left[n \cdot \left(k - \frac{1}{2} \right) \cdot \frac{\pi}{K} \right],\ n = 1, 2, ..., N_{cb}$. A diferença básica entre as variações é o tamanho e número de filtros cb. Na **variação 3** usamos 25 filtros cb triangulares igualmente espaçados como é mostrado na fig. 20, pelo próprio princípio de construção dos filtros cb, temos que K tem que ser referente a Fs/2 ou menor. Na **variação 4** usamos 30 filtros

cb triangulares, na **variação 5** ao invés de fixarmos o número de filtros, fixamos o a largura de banda em mels dos filtros cb, no caso usamos 300 mels de largura de banda. Nas variações anteriores usamos um liftro PB com pesagem para obtermos o número de coeficientes desejados. A **variação 6** é idêntica a variação 5 só que usamos um liftro PB retangular ao invés de um liftro PB com pesagem. Novamente não conseguimos obter resultados melhores que o dos coeficientes cepstrais.

Como até então não conseguimos implementar coeficientes mel-cepstrais que tivessem melhores resultados que os coeficientes cepstrais, comecei a questionar a eficiência dos coeficientes mel-cepstrais e para verificar tal eficiência, realizamos uma implementação com a função que gera os coeficientes mel-cepstrais (mcespt.m) da Voicebox [6]. A função que gerou estes coeficientes já estava pronta e seus resultados estão na tabela referente a **variação 7 (usamos 25 filtros cb triangulares)**. Esta tabela é tida como um prova da existência de coeficientes que nos geram resultados melhores que os cepstrais e da eficiência dos coeficientes mel-cepstrais.

As **variações de 8 a 12 (usamos 25 filtros cb triangulares)** são relativas a implementação apresentada no livro de Rabiner [3] porém, o livro não deixa claro como deve ser realizada a filtragem do espectro de potência pelos filtros cb. Abaixo temos alguns esclarecimentos que devem ser apresentados antes de continuarmos pois, mostram as diferentes formas de filtragem implementada neste trabalho e suas conseqüências.

- **cb** é a matriz dos filtros cb, de tamanho **cb(1:nf,1:K)** onde **nf** é o número de filtros cb usados e **K** é a maior freqüência de cb cujo valor é não nulo, ou seja, extremo direito do cb(nf,…) não nulo.
- antes de fazer o log de qualquer valor nestes procedimentos, nós testamos para ver se o valor é zero, caso afirmativo então substituímos por um valor na casa de 10^{-6} para não dar erro relativo ao logaritmo de zero, que é menos infinito.
- **Variações 8 e 9 - sum(log(cb.*r_3))** resulta na filtragem do espectro de potência de um segmento pelos nf filtros cb, individualmente. É a multiplicação elemento a elemento de **cb(1,1:K).*r_3(1,1:K)**, sendo **r_3(j,1:K)=r_3(1,1:K)=r(1:K) (r(1:K) é o espectro de potência de s(n))**, fazemos o log de cada elemento desta matriz e depois fazemos o somatório em colunas, ou seja, **log[cb(1,1)*r_3(1,1)]+log[cb(2,1)*r_3(2,1)]+…+ log[cb(nf,1)*r_3(nf,1)]** e assim por diante para cada coluna. Este somatório é igual a **[log(cb(1,1))+log(cb(2,1))+…+ log(cb(nf,1))]*log(r(1))**.
- **Variação 10 -** ao fazermos **cb*r**, estamos filtrando individualmente o espectro de potência do segmento por cada filtro cb e fazendo o somatório dos elementos de cada filtragem, depois faremos a **DCT** (Matlab) no log do resultado desta filtragem (procedimento semelhante ao usado pelo algoritmo proposto no livro de Deller [1]).
- **Variações 11 e12 - log(sum(cb.*r_3))** resulta na filtragem de um segmento pelos nf filtros cb, individualmente. É a multiplicação elemento a elemento de **cb(1,1:K)*r_3(1,1:K)**, sendo **r_3(j,1:K)=r_3(1,1:K)=r(1:K) (r(1:K) é o**

espectro de potência de s(n)), depois fazemos o somatório em colunas, ou seja, **cb(1,1)*r_3(1,1)+cb(2,1)*r_3(2,1)+…+cb(nf,1)*r_3(nf,1)** e assim por diante para cada coluna. Este somatório é igual a **[cb(1,1)+cb(2,1)+…+ cb(nf,1)]*r(1)**. Uma vez obtido este resultado então aplicamos o logaritmo. A única diferença entre as **variações 11 e 12**, é que na **11** usamos **cb(1,1)*r_3(1,1)+cb(2,1)*r_3(2,1)+…+cb(nf,1)*r_3(nf,1)** e na **12** usamos **[cb(1,1)+cb(2,1)+…+ cb(nf,1)]*r(1)=cbf(1)*r(1)**. O resultado na 12 é melhor devido a um número menor de erros procedente da substituição de zero por 10^{-6}. Na 11 fazemos esta substituição nos elementos de uma matriz (1:nf)x(1:K) e na 12 esta substituição é feita em um único vetor 1x(1:K) logo, o número de erros devido a substituição que podem ser inseridos numa matriz é bem maior do que o número que pode ser inserido num vetor.

Como podemos notar, a variação 12 nos fornece o melhor resultado e este também é obtido em um tempo de execução menor. A passagem da variação 11 para 12 tinha a princípio o intuito de tornar o algoritmo mais rápido porém como já explicamos acima, ela repercutiu num número menor de erros de substituição por conseguinte aumentando o percentual de acertos de reconhecimento.

Até então, o resultado da variação 12 é o melhor conseguido por nós neste trabalho, e se compararmos os tempos de execução de uma bateria de 100 sinais de teste temos:

- Coef. Cepstrais => aproximadamente **30min**.
- Coef. Mel-cepstrais (variação 11) => aproximadamente **43min**.
- Coef. Mel-cepstrais (variação 12) => aproximadamente **25min**.

Tendo em vista o resultado percentual do reconhecimento e o tempo de execução, a implementação utilizando os coeficientes mel-cepstrais da variação 12 é sem dúvida muito melhor que a implementação usando os coeficientes cepstrais ou qualquer implementação de coeficientes mel-cepstrais utilizada neste trabalho.

		Coef. Cepstrais	
		Universo de treinamento	Universo de Teste
1 centróide	Acertos:	92%	66%
	Erros:	8%	34%
	-	-	-
2 classes menores	Acertos:	96%	70%
	Erros:	4%	30%
	Dúvidas:	0%	0%
5 classes menores	Acertos:	96%	68%
	Erros:	4%	32%
	Dúvidas:	0%	0%

Esta **primeira tabela** nos fornece o resultado do teste realizado com os coeficientes cepstrais **(mesma tabela do teste de tamanho de frame usando 20 ms)**, e tem por intuito servir de padrão para a comparação com os testes a serem realizados com os coeficientes mel-cepstrais pois, o objetivo é comparar os resultados.

		Variação 0 (eq. Deller)		1ª - Variação	
		Universo de treinamento	Universo de Teste	Universo de treinamento	Universo de Teste
1 centróide	Acertos:	70%	46%	92%	60%
	Erros:	30%	54%	8%	40%
	-	-	-	-	-
2 classes menores	Acertos:	84%	58%	98%	64%
	Erros:	16%	42%	2%	36%
	Dúvidas:	0%	0%	0%	0%
5 classes menores	Acertos:	90%	58%	98%	68%
	Erros:	10%	42%	2%	32%
	Dúvidas:	0%	0%	0%	0%

		2ª - variação	
		Universo de treinamento	Universo de Teste
1 centróide	Acertos:	92%	58%
	Erros:	8%	42%
	-	-	-
2 classes menores	Acertos:	98%	56%
	Erros:	2%	44%
	Dúvidas:	0%	0%
5 classes menores	Acertos:	100%	56%
	Erros:	0%	44%
	Dúvidas:	0%	0%

- No primeiro resultado usamos integralmente a formulação dada pelo livro de Deller (fig. 21 deste trabalho).
- **1ª – variação**: retiramos o fator multiplicativo (2/N') do último bloco da fig. 21.
- **2ª – variação**: não utilizamos o último bloco da fig. 21 para o cálculo dos coef. Mel-cepstrais, e utilizamos a DCT (discrete cosine transform) do Matlab pois, esta é mencionada no livro de Rabiner e na referência [7] como sendo a última etapa no cálculo destes coeficientes.

Tabelas referentes ao **Experimento 4** – parte1.

		3ª - Variação		4ª - Variação	
		Universo de treinamento	Universo de Teste	Universo de Treinamento	Universo de Teste
1 centróide	Acertos:	74%	28%	76%	28%
	Erros:	26%	72%	24%	72%
	-	-	-	-	-
2 classes menores	Acertos:	88%	26%	88%	26%
	Erros:	12%	74%	12%	74%
	Dúvidas:	0%	0%	0%	0%
5 classes menores	Acertos:	90%	26%	90%	26%
	Erros:	10%	74%	10%	74%
	Dúvidas:	0%	0%	0%	0%

		5ª - Variação		6ª - Variação	
		Universo de treinamento	Universo de Teste	Universo de treinamento	Universo de Teste
1 centróide	Acertos:	74%	30%	74%	30%
	Erros:	26%	70%	26%	70%
	-	-	-	-	-
2 classes menores	Acertos:	88%	30%	88%	30%
	Erros:	12%	70%	12%	70%
	Dúvidas:	0%	0%	0%	0%
5 classes menores	Acertos:	90%	32%	90%	32%
	Erros:	10%	68%	10%	68%
	Dúvidas:	0%	0%	0%	0%

Tabelas referentes ao **Experimento 4** – parte2.

Nestas variações ao invés de utilizarmos S(k,m) como sendo o espectro de potência de um frame do sinal e não mais o espectro de um frame do sinal (primeiro bloco da fig. 21). Calculamos Y(i) e $\widetilde{Y}(k)$ normalmente dado pela fig. 21 e depois fazemos a DCT dada pela fórmula do Rabiner modificada

$$\widetilde{c}_n = \sum_{k=1}^{K} \left(\widetilde{Y}(k)\right) \cdot \cos\left[n \cdot \left(k - \frac{1}{2} \right) \cdot \frac{\pi}{K} \right], \; n = 1, 2, ..., N_{cb} .$$

Na fórmula real do livro de Rabiner ao invés de $\widetilde{Y}(k)$ usamos $\log(\widetilde{S}_k)$. Agora basta utilizarmos um lifter PB com pesagem para "pegarmos" o número desejado de coeficientes por frame.

- **3ª – variação:** usamos 25 filtros cb igualmente espaçados com 50% de superposição em mels. A freqüência máxima do ponto de maior freqüência dos filtros é Fs/2.
- **4ª – variação**: usamos 30 filtros cb igualmente espaçados com 50% de superposição em mels.
- **5ª – variação:** usamos 300 mels de largura de banda dos filtros cb e 150 mels de superposição.
- **6ª – variação:** usamos igualmente 300 mels de largura de banda dos filtros cb e 150 mels de superposição porém o liftro PB será usado sem pesagem.

		7ª - Variação	
		Universo de treinamento	Universo de Teste
1 centróide	Acertos:	98%	76%
	Erros:	2%	24%
	-	-	-
2 classes menores	Acertos:	100%	82%
	Erros:	0%	18%
	Dúvidas:	0%	0%
5 classes menores	Acertos:	100%	82%
	Erros:	0%	18%
	Dúvidas:	0%	0%

7ª - variação: Como até então não conseguimos obter resultados melhores que os dos coef. Cepstrais, tentamos fazê-lo com a utilização da função mcepst.m da Voicebox [6], e podemos notar que o resultado superou bastante o resultado obtido com os coef. Cepstrais. Neste e em todos os teste subseqüentes usaremos 25 filtros cb igualmente espaçados.

		8ª - Variação		9ª - Variação	
		Universo de treinamento	Universo de Teste	Universo de treinamento	Universo de Teste
1 centróide	Acertos:	46%	26%	100%	76%
	Erros:	54%	74%	0%	24%
	-	-	-	-	-
2 classes menores	Acertos:	52%	82%	100%	82%
	Erros:	48%	18%	0%	18%
	Dúvidas:	0%	0%	0%	0%
5 classes menores	Acertos:	80%	24%	100%	82%
	Erros:	20%	76%	10%	18%
	Dúvidas:	0%	0%	0%	0%

		10ª - Variação		11ª - Variação	
		Universo de treinamento	Universo de Teste	Universo de treinamento	Universo de Teste
1 centróide	Acertos:	94%	62%	100%	80%
	Erros:	6%	38%	0%	20%
	-	-	-	-	-
2 classes menores	Acertos:	98%	64%	100%	80%
	Erros:	2%	36%	0%	20%
	Dúvidas:	0%	0%	0%	0%
5 classes menores	Acertos:	100%	70%	100%	84%
	Erros:	0%	30%	10%	16%
	Dúvidas:	0%	0%	0%	0%

Tabelas referentes ao **Experimento 4** – parte 3.

		12ª - Variação	
		Universo de treinamento	Universo de Teste
1 centróide	Acertos:	100%	82%
	Erros:	0%	18%
	-	-	-
2 classes menores	Acertos:	100%	84%
	Erros:	0%	16%
	Dúvidas:	0%	0%
5 classes menores	Acertos:	100%	86%
	Erros:	0%	14%
	Dúvidas:	0%	0%

Tabela referente ao **Resultado 4** – parte 3 – última tabela.

Nas **variações de 8 a 12** usaremos a implementação proposta no livro de Rabiner [3], através da eq. $\tilde{c}_n = \sum_{k=1}^{K} \left(\log(\tilde{S}_k)\right) \cdot \cos\left[n \cdot \left(k - \frac{1}{2} \right) \cdot \frac{\pi}{K} \right],\ n = 1, 2, ..., N_{cb}$, mais um liftro PB com pesagem.

- **8ª – variação:** Na fórmula acima usamos K = N', onde N' é o número de pontos usados no cálculo da DTFT da potência de s(n) e $\log(\tilde{S}_k) = \text{sum}(\log(cb. * r_3))$.
- **9ª – variação:** Na fórmula acima usamos K = a freq. máxima do filtro cb, que sabemos ser $\leq$ Fs/2 e $\log(\tilde{S}_k) = \text{sum}(\log(cb. * r_3))$.
- **10ª – variação:** Na fórmula acima usamos DCT (do Matlab) de $\log(\tilde{S}_k)$ ao invés dos cossenos multiplicados acima e $\log(\tilde{S}_k) = \log(cb * r)$.
- **11ª – variação:** Na fórmula acima usamos, $\log(\tilde{S}_k) = \log(\text{sum}(cb. * r_3))$.
- **12ª – variação:** Na fórmula acima usamos, $\log(\tilde{S}_k) = \log(cbf. * r)$.

Experimento 5: Teste do número de coeficientes mel-cepstrais (mc)

Usando:

- tframe=20 ms.
- janela (segmentação)=retangular.
- Sem superposição (segmentação).
- dtw11.
- **m** variável (8, 10 ou 12).
- Total de coef. por segmento é 1e+1de+1dde+(**m***mc)+(**m***dmc)+(**m***ddc).
- 25 filtros cb triangulares.
- Sem limiar.
- Usamos a implementação da **12ª variação** do **experimento 4**.

Neste teste variamos o valor de **m**, que é o número de coeficientes mel-cepstrais extraídos na análise cepstral. O valores de **m** testados serão: 8, 10 e 12.

O intuito é representar o segmento com o menor número possível de coeficientes, de modo que não prejudique o percentual de acertos de reconhecimento.

Observando os resultados notamos que não há mudança significativa entre a utilização de **m**=10 e **m**=12, e também foi constatado que todos os três teste tiveram um tempo de execução de aproximadamente **25 min**. Desta maneira, o que nos fornece melhor resultado é o que tem menor número de coeficientes e resultados bastante expressivos de reconhecimento, ou seja, a tabela onde **m = 10**.

		27mc =>m = 8		33mc = > m = 10	
		Universo de treinamento	Universo de Teste	Universo de treinamento	Universo de Teste
1 centróide	Acertos:	100%	78%	100%	82%
	Erros:	0%	22%	0%	18%
	-	-	-	-	-
2 classes menores	Acertos:	100%	80%	100%	84%
	Erros:	0%	20%	0%	16%
	Dúvidas:	0%	0%	0%	0%
5 classes menores	Acertos:	100%	86%	100%	88%
	Erros:	0%	14%	0%	12%
	Dúvidas:	0%	0%	0%	0%

		39mc => m =12	
		Universo de Treinamento	Universo de Teste
1 centróide	Acertos:	100%	82%
	Erros:	0%	18%
	-	-	-
2 classes menores	Acertos:	100%	84%
	Erros:	0%	16%
	Dúvidas:	0%	0%
5 classes menores	Acertos:	100%	86%
	Erros:	0%	14%
	Dúvidas:	0%	0%

Tabelas referentes ao **Experimento 5**.

Experimento 6: Teste do número de coeficientes delta

Usando:
- tframe=20ms.
- janela (segmentação)=retangular.
- Sem superposição (segmentação).
- dtw11.
- **m** = 12.
- Total de coef. por segmento é variável:
  - 1e+12mc.
  - 1e+1de+12*mc+12*dmc.
  - 1e+1de+1dde+12*mc+12*dmc+12*ddmc.
- 25 filtros cb triangulares.
- Sem limiar.
- Usamos a implementação da **12ª variação** do **experimento 4**.

Neste teste continuamos com o intuito de diminuir o número de coeficientes até o menor número possível porém, não estamos alterando o valor de **m**, e sim o número de coeficientes delta. Na primeira tabela usamos somente a energia e 12 coeficientes mel-cepstrais para representar um segmento; na segunda tabela usamos energia, delta-energia, 12 coef. mel-cepstrais e 12 delta mel-cepstrais; e na terceira tabela usamos energia, delta-energia, delta-delta-energia, 12 coef. mel-cepstrais, 12 delta mel-cepstrais e 12 delta delta mel-cepstrais.

Mesmo não observando diferenças muito marcantes nos resultados, creio que o resultado mais razoável é aquele que utiliza energia, delta-energia, 12 mel-cepstrais e 12 delta mel-cepstrais.

O resultado deste teste deve ser combinado com o resultado do teste anterior, deste modo, conseguiremos obter o menor número possível de coeficientes para a representação de cada segmento do sinal. Vale a pena relatar que as três tabelas tiveram aproximadamente o mesmo tempo de execução para os 100 sinais **(25 min)**.

		e+12mc		e+de+12mc+12dmc	
		Universo de treinamento	Universo de Teste	Universo de treinamento	Universo de Teste
1 centróide	Acertos:	100%	80%	100%	82%
	Erros:	0%	20%	0%	18%
	-	-	-	-	-
2 classes menores	Acertos:	100%	84%	100%	82%
	Erros:	0%	16%	0%	18%
	Dúvidas:	0%	0%	0%	0%
5 classes menores	Acertos:	100%	84%	100%	86%
	Erros:	0%	16%	0%	14%
	Dúvidas:	0%	0%	0%	0%

		e+de+dde+12mc+12dmc+12ddmc	
		Universo de Treinamento	Universo de Teste
1 centróide	Acertos:	100%	82%
	Erros:	0%	18%
	-	-	-
2 classes menores	Acertos:	100%	84%
	Erros:	0%	16%
	Dúvidas:	0%	0%
5 classes menores	Acertos:	100%	86%
	Erros:	0%	14%
	Dúvidas:	0%	0%

Tabelas referentes ao **Experimento 6**.

Experimento 7: Teste do melhor desempenho do sistema total.

Neste teste analisaremos o resultado tendo como universo de treinamento sinais de fala recortados, não importando o quão bem feitos são estes recortes **(ver seção 2.4 – Tabela 8)**, pois, nos testes anteriores usamos o universo de treinamento de sinais bem recortados.

Utilizaremos também todos os melhores resultados obtidos com os teste anteriores realizados. Na verdade, faremos testes que tentem otimizar o resultado num menor tempo de execução, considerando os resultados obtidos nos testes feitos anteriormente.

Para os testes usaremos:

- Coeficientes mel-cepstrais implementado na **12ª variação do experimento 4**.
- **m**=10.
- 20mc => 1e+1de+10*mc+10*dmc.
- 25 filtros cb triangulares.
- sem superposição (segmentação).
- janela retangular (segmentação).
- usamos como limiar (1,5)*dmax.

Começaremos na **tabela #1** usando como universo de treinamento os sinais bem recortados utilizados nos testes anteriores. Este servirá de referência para os testes subsequentes que tem por intuito medir a eficiência do sistema total.

Tabela #1

- universo de treinamento bem recortado, os mesmos sinais usados anteriormente.
- dtw11.
- tframe=20 ms.

Tabela #2

- universo de treinamento não necessariamente bem recortado, nem todos os recortes ficaram bem feitos.
- dtw11.
- tframe=20 ms.

Tabela #3

- universo de treinamento igual ao anterior (tabela #2).
- dtw11.
- tframe=15 ms.

Tabela #4

- universo de treinamento igual ao anterior (tabela #2).
- dtw22_4.
- tframe=20 ms.

		#1	#2
		Universo total	Universo total
1 centróide	Acertos:	89%	81%
	Erros:	5%	16%
	-	-	-
	Nr:	6%	3%
2 classes menores	Acertos:	90%	89%
	Erros:	4%	8%
	Dúvidas:	0%	0%
	Nr:	6%	3%
5 classes menores	Acertos:	91%	93%
	Erros:	3%	4%
	Dúvidas:	0%	0%
	Nr:	6%	3%
	Tempo	**23min**	**25min**

Na **tabela #1** testamos com os sinais de treinamento com 100% acerto no recorte e na **#2** nem todos os sinais tem o recorte perfeito.
O resultado usando somente 1 centróide caracteriza bem as nossas expectativas. No universo de treinamento usado em #1 consideramos somente as imperfeições do sistema de reconhecimento; e em #2 consideramos as imperfeições do sistema total.

		#3	#4
		Universo total	Universo Total
1 centróide	Acertos:	76%	79%
	Erros:	21%	19%
	-	-	-
	Nr:	3%	2%
2 classes menores	Acertos:	86%	89%
	Erros:	11%	9%
	Dúvidas:	0%	0%
	Nr:	3%	2%
5 classes menores	Acertos:	89%	92%
	Erros:	8%	6%
	Dúvidas:	0%	0%
	Nr:	3%	2%
	Tempo	**70min**	**70min**

Na **tabela #3** não obtivemos resultados tão bons quanto esperávamos, já que, no experimento 2 tivemos melhores resultados com segmento de tamanho igual a 15 ms.
Na **tabela #4** o resultado também não foi muito melhor que o da tabela #2 e o custo computacional aumentou assustadoramente.

O teste realizado na tabela #2 é o de melhor resultado, desta maneira, podemos avaliar o grau de eficiência do processamento geral do sistema. Apresentaremos a seguir uma tabela com os resultados da tabela #2, só que detalhados por universo de treinamento e teste, separadamente.

		#2 (detalhada)	
		Universo de treinamento	Universo de teste
1 centróide	Acertos:	86%	76%
	Erros:	14%	18%
	-	-	-
	Nr:	0%	6%
2 classes menores	Acertos:	90%	80%
	Erros:	10%	6%
	Dúvidas:	0%	0%
	Nr:	0%	6%
5 classes menores	Acertos:	96%	90%
	Erros:	4%	4%
	Dúvidas:	0%	0%
	Nr:	0%	6%

6 – Conclusões

De forma geral, os resultados foram bastante satisfatórios, pois, o intuito do nossos trabalho era basicamente de cunho didático. O trabalho poderá servir de ferramenta para futuros alunos do grupo de processamento de sinais de fala da UFRJ que venham a trabalhar com reconhecimento e para todos que tiverem interesse e desejam se aprofundar nesta área.

Comprovamos a maior eficiência dos coeficientes mel-cepstrais sobre os cepstrais na representação de sinais de fala e observamos diferenças entre a algumas implementações encontradas na literatura. Fizemos vários testes com as diferentes implementações e chegamos ao resultado mostrado no experimento 4, onde obtivemos resultados realmente muito satisfatórios.

No último experimento fazemos um teste final com os "melhores" resultados de cada experimento e com sinais de teste não tão bem recortados. O objetivo deste experimento era testar a eficiência do sistema como um todo e não mais a eficiência dos blocos em separado. Neste teste tentamos usar resultados razoáveis, ou seja, uma boa relação entre tempo de execução e percentual de acertos. Assumindo estas considerações, os parâmetros usados foram:

- A implementação dos coeficientes mel-cepstrais usado na **12ª variação** do **experimento 4**.
- **m=10** (número de coeficientes extraídos por segmento).
- **20mc** => 1e+1de+10*mc+10*mc (20 parâmetros por segmento, sendo um com o valor de energia, um delta-energia, 10 coeficientes mel-cepstrais e mais 10 delta mel-cepstrais.
- **25** filtros cb **triangulares**.
- **Não usamos superposição** na segmentação do sinal.
- Segmentamos o sinal com uma janela **retangular**.
- Usamos **(1,5)*dmax** como limiar de reconhecimento.
- **dtw11** como restrição local de caminho no algoritmo DTW.
- Tamanho de segmento de **20 ms**.

O sistema de reconhecimento utilizado neste trabalho é bastante limitado, pois considera um vocabulário bem pequeno e é dependente do interlocutor. Este sistema também não funciona em tempo real, uma vez que a ferramenta utilizada em sua implementação foi o Matlab (suas desvantagens foram apresentadas no decorrer do trabalho), porém se considerarmos os resultados como sendo uma primeira etapa para futuros sistemas de reconhecimentos, que virão a ser implementados pelos alunos do grupo de processamento de sinais de fala da UFRJ, estes podem ser considerados como um grande passo rumo a implementação de sistemas mais eficientes e mais robustos.

Devido a existência de "infinitas" possibilidades de teste que ainda podem ser feitas com os algoritmos implementados neste trabalho e o interesse do grupo em realizar trabalhos cada vez mais sofisticados nesta área,

iremos concentrar os nossos esforços futuros na implementação de sistemas de reconhecimento utilizando a técnica de HMM, de modo que, possamos utilizar vocabulários maiores e independência de interlocutor. Este direcionamento futuro estará dando continuidade e aproveitará os resultados obtidos neste trabalho.

7 – Bibliografia

[1] J. R. Deller, J. G. Proakis e J. H. Hansen, *Discrete-Time Processing of Speech Signals*. MacMillan, 1993.
[2] S. K. Mitra, *Digital Signal Processing: A Computer-Based approach.* McGraw-Hill, 1998.
[3] L. Rabiner e B. Juang, *Fundamentals on Speech Recognition*. Prentice Hall, 1996.
[4] S. Furui, *Digital Speech Processing, Synthesis, and Recognition*, Marcel Dekker, Inc, 1989.
[5] S. Haykin, *An Introduction to analog and digital communications*, J. Wiley & Sons, 1989.
[6] Voicebox obtida pela internet do Imperial College.
http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html
[7] C. Espain, Comunicação Pessoal – Universidade do Porto.
[8] T. Robinson, Speech Analysis: Lecture notes, obtidas pela internet.
http://svr-www.eng.cam.ac.uk/~ajr/SpeechAnalysis/index.html
[9] Site do comp.speech Frequently asked questions.
http://www.itl.atr.co.jp/comp.speech/
[10] J. Trimble III, Front end parametrization: Linear prediction based measurements, Course project, Mississippi State University.
http://www.isip.msstate.edu/publications/courses/ece_8993_speech/conference/1996/
[11] R. Duncam, M. Vishwanath, Y. Wu e J. Zhao, Implementation and analysis of Speech Recognition Front-Ends, Course project, Mississippi State University.
http://www.isip.msstate.edu/publications/conferences/ieee_secon/1999/front_end/

[8]