JUSSARA RIBEIRO

“SÍNTESE COMPUTACIONAL DE VOZ”

Trabalho de Conclusão de Curso apresentado à Escola de Engenharia de São Carlos, da Universidade de São Paulo

Curso de Engenharia Elétrica com ênfase em Eletrônica

ORIENTADOR: Rodrigo Capobianco Guido

São Carlos

2010

Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca – EESC/USP

R484s

Ribeiro, Jussara
Síntese computacional de voz / Jussara Ribeiro ; orientador Rodrigo Capobianco Guido. -- São Carlos, 2010.

Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2010.

1. Síntese de voz. 2. Áudio digital. 3. Algoritmos. 4. Fisiologia oral. I. Título.

DEDICATÓRIA

Dedico este trabalho primeiramente a Deus, pois sem Ele, nada seria possível.

Aos meus pais José Marcos e Lezira e a minha irmã Juliana, que sempre me incentivaram e estiveram ao meu lado em todas as horas, compartilhando os momentos de felicidade e tristeza.

Aos meus grandes amigos Sabrina, Ana Paula e Michel e ao meu namorado Frederico, pela amizade e companheirismo ao longo desta jornada.

Dedico este trabalho a toda minha família e em especial, a meu avô José Alfeu, pois seu apoio foi fundamental na realização deste sonho.

AGRADECIMENTOS

Agradeço a todas as pessoas que contribuíram, diretamente ou indiretamente, para realização deste trabalho.

Aos professores, que contribuíram para meu desenvolvimento pessoal e profissional.

Em especial, agradeço ao Professor Rodrigo Capobianco Guido, pela oportunidade, por sua atenção, incentivo e ajuda.

Conteúdo

Índice de Figuras

Resumo

Este trabalho tem como objetivo desenvolver um programa para a síntese computacional de voz, baseado na excitação glotal artificial e no uso de filtros, que simulam o trato vocal humano. Utilizando o modelo fonte-filtro, que caracteriza o mecanismo de produção de voz, é possível desenvolver um algoritmo para gerar a excitação e o filtro em duas etapas distintas e desta forma obter o sinal desejado.

Palavras-Chave: Síntese de Voz, excitação artificial, modelo fonte-filtro, fisiologia da fala, algoritmo, filtro.

Abstract

This work aims at creating an algorithm for speech synthesis, based on artificial glottal excitation and, then, its filtering, in order to simulate the human vocal tract. By using a source-filter model, which characterizes the voice production mechanism, it is possible to develop an algorithm to produce the excitation and filtering, in individual steps, so that the intended signal is obtained.

Keys Word: Speech synthesis, artificial excitation, source-filter model, physiology of speech, algorithm, filter.

1. Introdução

A síntese de voz é o processo de produção artificial de voz humana [1]. É uma especialidade pertencente à sub-área de processamento de sinais [2] das áreas de Engenharia Elétrica, Ciência da Computação e Física Aplicada [3].

O sistema de síntese de voz está em expansão e possui enorme aplicabilidade real em diversos campos, destacando as aplicações em telecomunicação e multimídia.

O objetivo fundamental do presente trabalho é o de pesquisar os algoritmos existentes para a síntese de voz, com o estudo detalhado do mecanismo de geração do sinal de excitação *(voiced e unvoiced*) e o seu tratamento pelo sistema vocal, implementando um sistema simples de síntese de voz.

A implementação, na sua totalidade ou em parte, utiliza um computador pessoal comum com sistema operacional LINUX e linguagem de programação C/C++, podendo ser realizada em tempo-real com o uso de um *Digital Signal Processor* (DSP) [4], sendo possível aplicar o algoritmo na área de autenticação biométrica [5] e interpretação de comandos por voz [6].

2. Revisão bibliográfica

De acordo com o cronograma inicialmente proposto, durante a primeira fase do projeto iniciou-se a pesquisa de algoritmos existentes para síntese computacional de voz e, além disso, buscou-se conhecer e entender o processo de geração natural de voz. Dessa forma, a revisão bibliográfica realizada possibilitou o entendimento dos conceitos explicados a seguir.

2.1. Fisiologia da fala

No início da história evolutiva, os órgãos de fonação eram responsáveis por executar as funções necessárias para preservação da vida: respiração e alimentação, e não tinham a função de produzir voz. Posteriormente, desenvolveu-se a capacidade de produção de voz, o que adaptou todas essas estruturas a uma nova função [7]. A voz humana é a principal ferramenta de comunicação e é resultado de um complexo processo, cujo mecanismo envolve o sistema nervoso central, o sistema fonador e o sistema respiratório.

Apesar de ser um mecanismo repleto de detalhes, é possível entender a fisiologia do aparato vocal realizando uma divisão do mesmo em subsistemas, destacando as características anatômicas mais relevantes na acústica das vogais, conforme apresentado na figura1:

- Respiratório: constituído pelos pulmões, músculos respiratórios, brônquios e a traquéia, sendo responsável pela geração da energia utilizada na produção de voz.
- Laríngeo: constituído por um conjunto de músculos, ligamentos e cartilagens sendo responsável pelo controle das pregas vocais (fonação).
- Supralaríngeo: composto pelas regiões faringal, bucal e nasal é responsável pela modulação do som gerado.

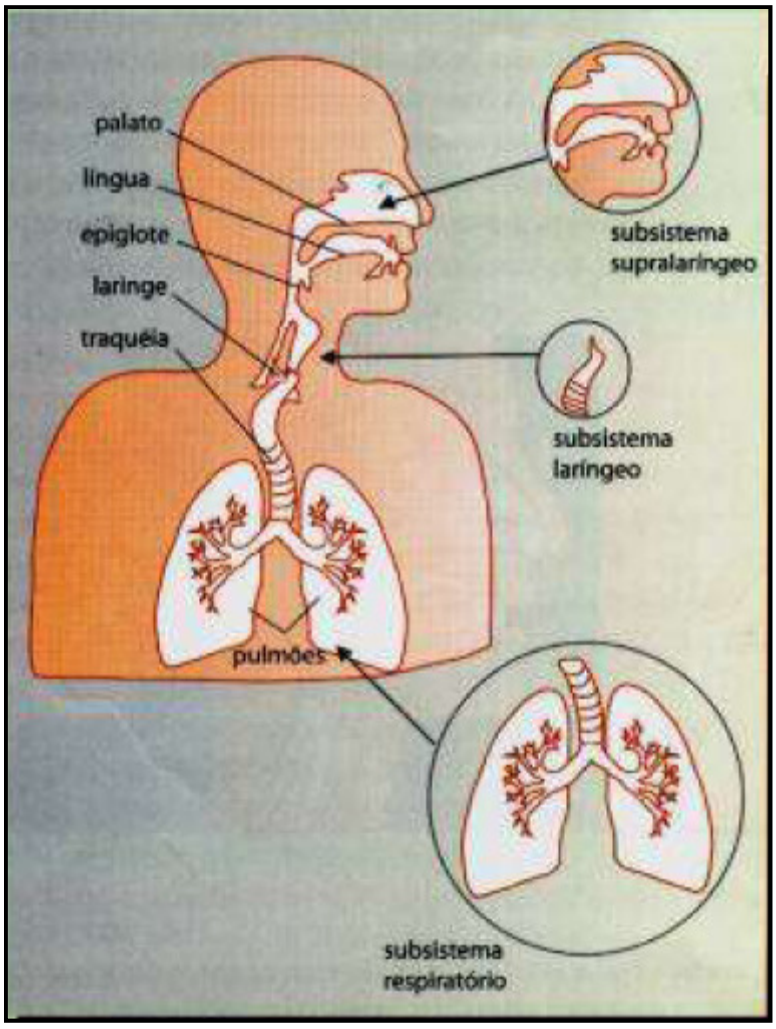

Figura 1: Sistema de produção de fala. (Fonte: http://revistaescola.abril.com.br)

2.2. Produção de voz

Na fala humana encontram-se sons produzidos de formas distintas. Portanto inicialmente o foco do estudo foi a produção de vogais. Utilizou-se o modelo fonte-filtro, que separa os fenômenos acústicos em três grupos independentes: a fonte sonora, o filtro acústico e a irradiação. Consiste basicamente da propulsão de ar pelos pulmões, seguida de um processo de filtragem realizado pelo trato vocal e elementos associados [8]. A interpretação física do sistema pode ser visualizada na figura 2.

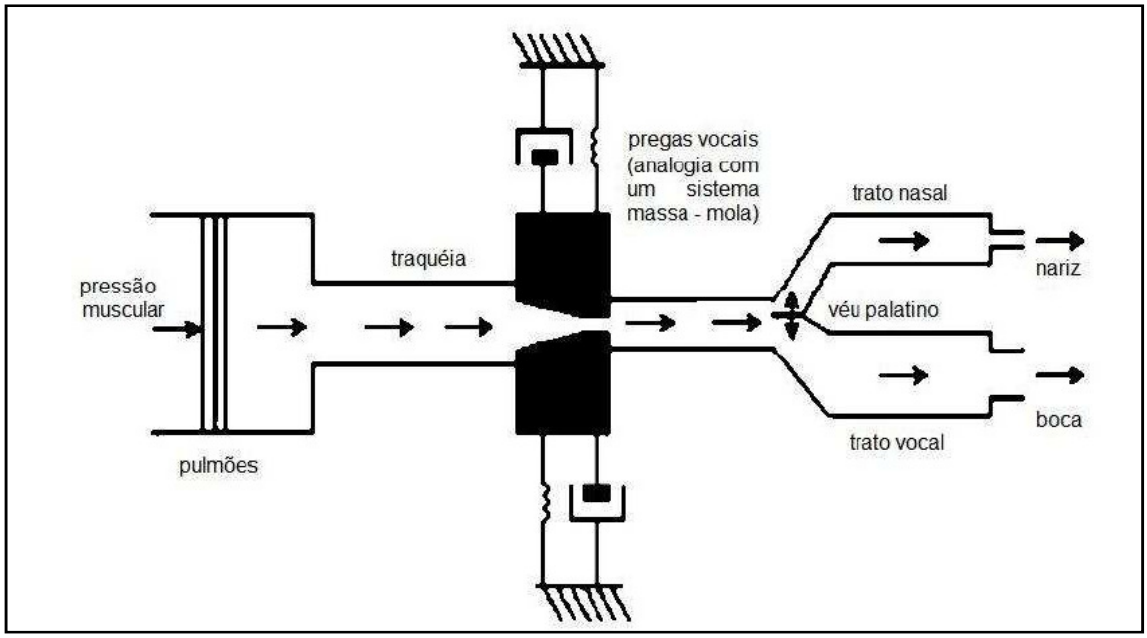

Figura 2: Interpretação física simplificada do sistema gerador de voz. (Fonte: Vieira, Lucimar S. Conversão de Voz Baseada na Transformada Wavelet. São Carlos: IFSC –USP,2007. Dissertação Mestrado).

O pulmão pode ser considerado uma fonte propulsora de ar que torna possível a produção de voz. Esta massa de ar é conduzida até a laringe pela traquéia, onde são produzidos os pulsos glotais para a produção de voz. A faringe age como um ressonador.

As pregas vocais controlam o fluxo de ar fornecido pelos pulmões, fazendo com que este sinal de excitação seja periódico, vibrando em determinada freqüência, ou aperiódico, similar a um sinal ruidoso. Se o sinal for periódico, este período é chamado de período de *pitch* e a voz produzida será classificada como *voiced speech*, caso contrário a voz será classificada como *unvoiced speech.*

Voiced speech são basicamente as vogais, foco deste trabalho. E *unvoiced speech* são os demais sons.

O trato vocal é composto por elementos que guardam íntima relação entre si, o que permite a modelagem do som e a projeção do mesmo no ambiente. Dependendo de como as estruturas seguintes às pregas vocais agem, pode-se ainda refinar esta classificação dos sinais de voz da seguinte forma [9]:

As consoantes são fonemas produzidos através da obstrução do ar proveniente do pulmão, precisando de uma vogal para ser emitidos. Esses obstáculos podem ser totais ou parciais, a partir da posição da língua e dos lábios.

As consoantes apresentam quatro critérios de classificação:

- modo de articulação: responsável pela identificação do obstáculo que ocorre durante a passagem do ar pela boca.

Se a corrente de ar encontrar um obstáculo total, essas consoantes serão classificadas como oclusivas (p, b, t, d, k e g).

Se o obstáculo for parcial, as consoantes serão chamadas constritivas (compressão), podendo ser fricativas (fricção do ar através de uma fenda no meio da boca), laterais (o ar sai pelos lados da boca) e vibrantes (quando ocorre a vibração da língua ou do véu palatal).

A classificação das consoantes constritivas ocorre da seguinte maneira:

- constritivas fricativas: f, v, s, z, x, j;

- constritivas laterais: l, lh;

- constritivas vibrantes: r, rr

- ponto de articulação: identifica em qual ponto da cavidade bucal localiza-se o obstáculo para a passagem do ar.

O ponto de articulação classifica-se em consoantes bilabiais (contato entre os lábios superior e inferior), labiodentais (o lábio inferior tem contato com os dentes incisivos superiores), linguodentais (contato entre a língua e a face interna dos dentes incisivos superiores), alveolares (contato da nvlíngua com os alvéolos dos dentes incisivo superiores), palatais (o dorso da língua toca o céu da boca) e velares (parte posterior da língua tem contato com o véu palatino).

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

Outros conceitos necessários para compreensão da produção de voz, utilizando o modelo fonte-filtro são:

- *freqüência fundamental, freqüência glótica (pitch):* é a componente de freqüência da excitação pulmonar, controlada pela vibração das cordas vocais no caso de sons vozeados (*voiced*). Ou seja, a freqüência glótica é o número de ciclos vibratórios completos da mucosa das pregas vocais por um segundo, sendo que quanto mais ciclos por segundo, mais alta será a freqüência e mais agudo o sinal de voz. Pode-se observar que a freqüência fundamental esta relacionada fortemente com o gênero e a idade, porém é considerada um dos parâmetros acústicos mais robustos. Os homens apresentam freqüência fundamental (fo) de, aproximadamente, 113 Hz e as mulheres 204 Hz e as crianças 300Hz, aproximadamente. [10]

- *formantes*: O filtro oral é caracterizado por picos (F1, F2, F3 e F4) no espectro de freqüências. Estes picos, que correspondem aos modos normais dos tubos acústicos, são tradicionalmente chamados de formantes. Os formantes mais importantes para o reconhecimento de uma vogal são os três primeiros (F1, F2, F3). Estes três primeiros formantes têm menor dependência com o locutor e prestam-se, principalmente, para diferenciar as vogais. As freqüências dos formantes, particularmente as duas primeiras (F1 e F2), dependem do formato do trato vocal entre a glote e os lábios. O terceiro formante, F3, está relacionado com a ressonância do restante do trato vocal com a passagem da constrição de língua. F4 é altamente relacionado ao timbre vocal, ou seja, ao componente pessoal do som vocal. Os formantes superiores (F4, F5, etc.), tem menor conteúdo lingüístico e maior variação com o locutor.

Devido a assimetria dos pulsos, o espectro possui além da freqüência fundamental, uma série harmônica onde as freqüências são múltiplos inteiros de fo, ou seja, tem-se uma onda rica em harmônicos.

A irradiação pode ser comparada a um filtro passa-alta, pois se observa que a pressão sonora numa região distante do locutor tende a ter maior intensidade relativa nas altas freqüências.

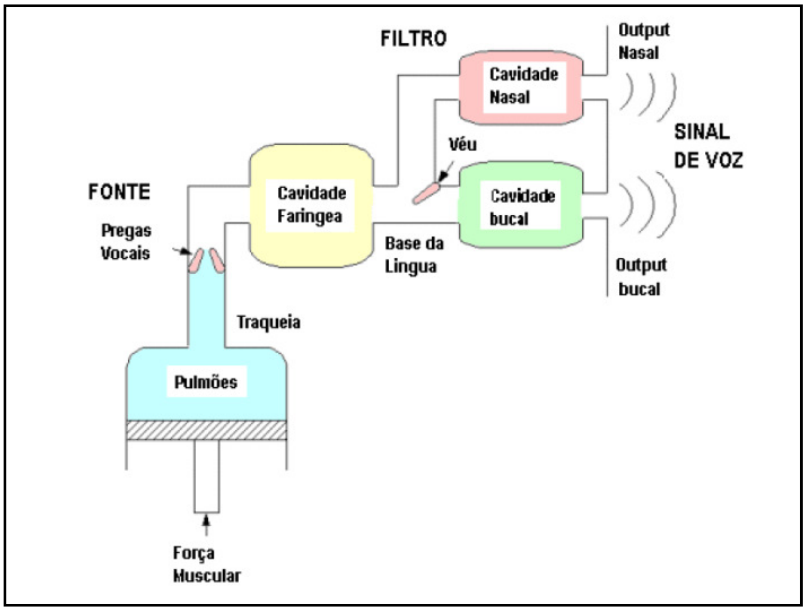

Figura 3: Sistema fonte-filtro. (Fonte: Dajer, Eugenia M. Padrões Visuais de Sinais de Voz Através de Técnicas de Análise Não Linear. São Carlos: EESC – USP, 2006. Dissertação (Mestrado).

2.3. Síntese de Voz

A ciência da fala apresenta característica multidisciplinar, abrangendo diversas áreas do conhecimento humano.

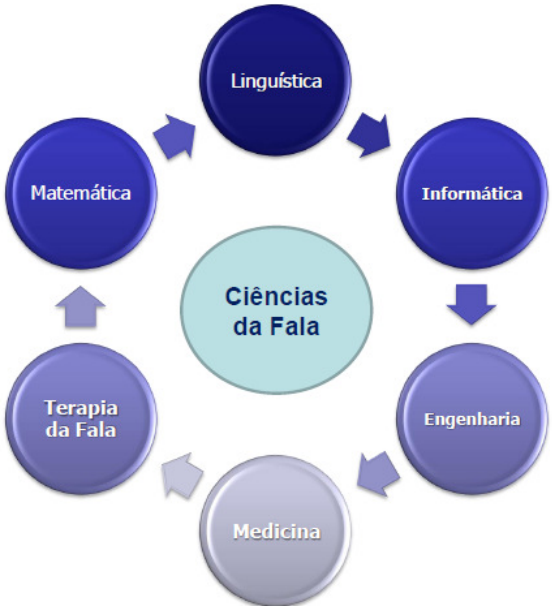

Figura 4: Ciência da fala.

Hoje, a síntese de voz pode ser aplicada em sistemas de orientação e navegação (sistemas de voz aplicados a navegação por GPS), no ensino e-learning com interfaces de voz, controle de sistema por voz e em telecomunicação (leitura SMS por voz, útil a cegos).

O interesse pela produção artificial de voz é uma curiosidade antiga. Em 1779, foi desenvolvido o primeiro projeto que tinha como objetivo a produção de voz artificial. Ele foi elaborado por Christian Gottlieb Krazenstein da Academia Imperial de St Petersburg. O trabalho consistiu na invenção de um instrumento que usava uma palheta vibrante e um constante fluxo de ar e o objetivo era produzir de forma artificial as vogais. Posteriormente, Wolfgang Von Kempelen, em 1791, após dedicar vinte anos de sua vida à criação da máquina falante, criou uma máquina manualmente operada utilizando um fole e uma palheta, que representavam o pulmão e as cordas vocais, respectivamente. Sua invenção foi descrita detalhadamente em seu livro, lançado em Viena. Esta máquina (exposta em Deutsches Museum de Munique e operacional até hoje) foi reproduzida várias vezes no século XIX, comprovando o interesse do homem em conhecer o mecanismo de produção da voz a partir de um modelo mecânico [11].

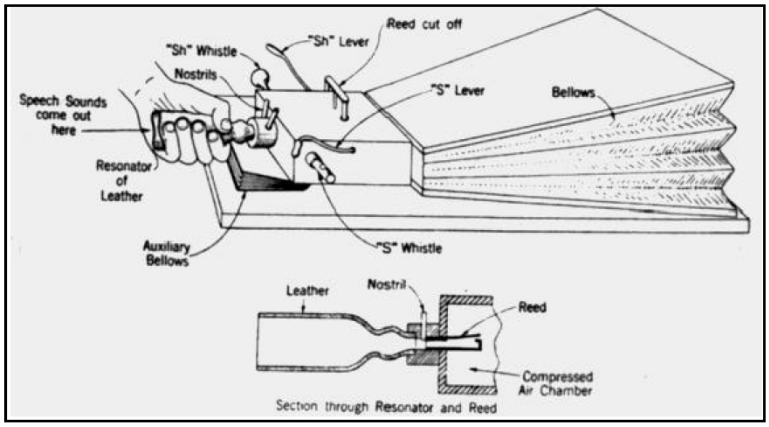

Figura 5: Máquina de Kempelen. (Fonte: Friedrich, Elcio. Sistema de Síntese de Voz Microcontrolado Portátil. Curitiba: Centro universitário Positivo – UnicenP, 2004.)

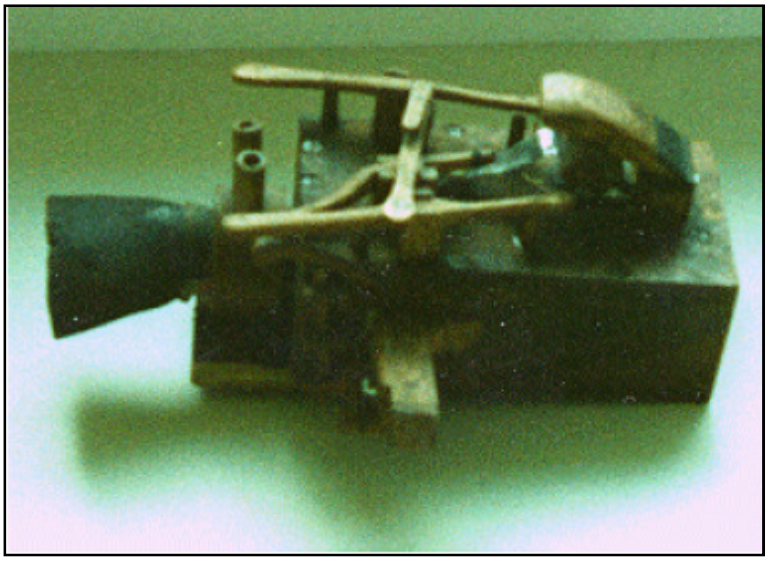

Figura 6: Foto da Máquina de Kempelen. (Fonte; http://www.unicamp.br/iel/site/docentes/LinInstLing.pdf)

Kempelen sintetiza aquilo que é mais relevante no funcionamento do aparelho fonador, reconhecendo assim o que se denomina os três sistemas de produção de fala: Respiratório (pulmão), laríngeo (sendo as pregas vogais a principal estrutura) e o supralaríngeo (trato vocal).

A máquina de Joseph Faber, Euphonia, foi construída em 1835 e demonstrada em Londres em 1846. Nesta máquina, Joseph fez um novo progresso incluindo no seu mecanismo de produção de fala um modelo de língua e uma cavidade laríngea cuja forma podia ser controlada. A máquina já podia ser operada com um teclado e tinha controle do tom laríngeo através de um pedal.

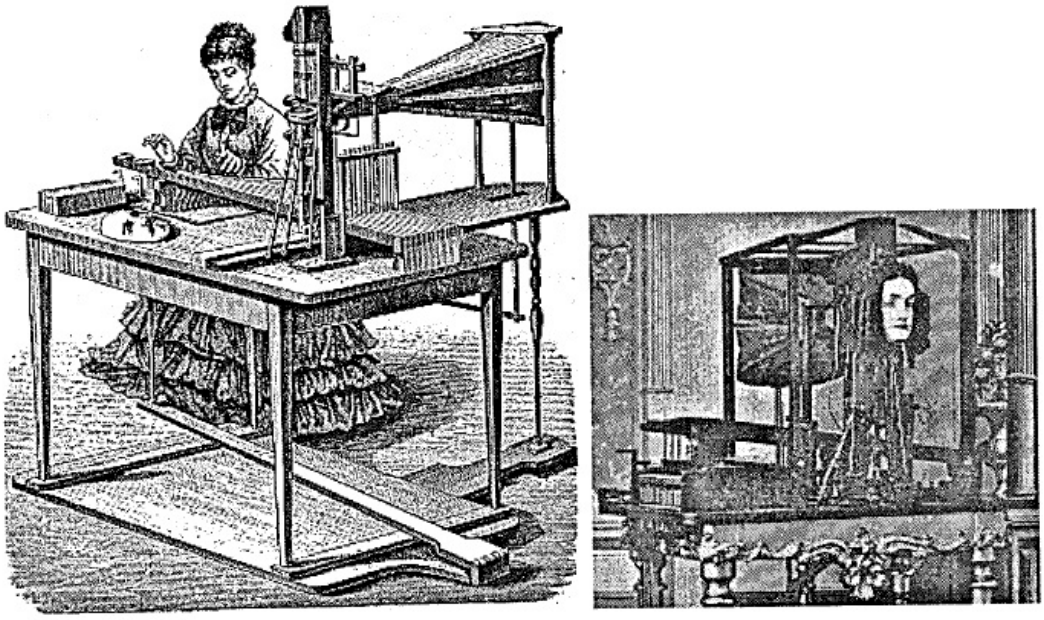

Figura 7: A Fabulosa Máquina Falante, de Joshep Faber. (Fonte: Barros, Maria J. A. Estudo Comparativo e Técnicas de Geração de Sinal para a Síntese da Fala. Porto: Faculdade de Engenharia da Universidade do Porto. 2002. Tese (Mestrado).

No século XX, com raras exceções, as tentativas de construção de máquinas falantes não mais procuravam dispor de engrenagens e outros dispositivos mecânicos, mas simular a produção de som através de equivalentes elétricos constituídos por circuitos dotados de resistores, capacitores e indutores. Tinha início a Era Elétrica das Máquinas Falantes.

Em 1922, Stewart foi o responsável pelo surgimento do primeiro dispositivo elétrico capaz de gerar alguns sons de fala sintética. Este dispositivo consistia de dois circuitos ressoadores excitados por um sinal sonoro de entrada: ajustando-se as freqüências de ressonância dos dois circuitos, podia-se simular o som de cada uma das vogais, desde que as freqüências de ressonância se aproximassem das freqüências dos dois primeiros formantes da vogal correspondente.

No entanto, o primeiro grande marco na história da síntese de voz ocorreu sem dúvida no ano de 1939. Nesta ocasião, o engenheiro Dudley do *Bell Laboratories* apresentou à comunidade científica o sintetizador de fala por ele desenvolvido, denominado VODER [12]. A publicação cientifica deste evento foi realizada na revista *"Science News Letter"*, na edição de 14 de janeiro desse mesmo ano. O sistema constituído de dois geradores de sons independentes (excitação), sendo um sistema responsável pela produção dos sons periódicos (*voiced*) e o outro responsável pela

geração de sons aperiódicos (*unvoiced*). Era manualmente controlado e basicamente tratava-se de um sintetizador espectral, com dez filtros passa-banda em paralelo, que abragiam todo o espectro de freqüências da fala. Todos os filtros eram excitados ou pela fonte de ruído ou por um oscilador. O controle do pitch era feito no oscilador, através de um pedal. As saídas dos filtros passavam por potenciômetros operados manualmente. Havia três teclas adicionais para excitar filtros selecionados, de modo a simular os sons das consoantes plosivas.

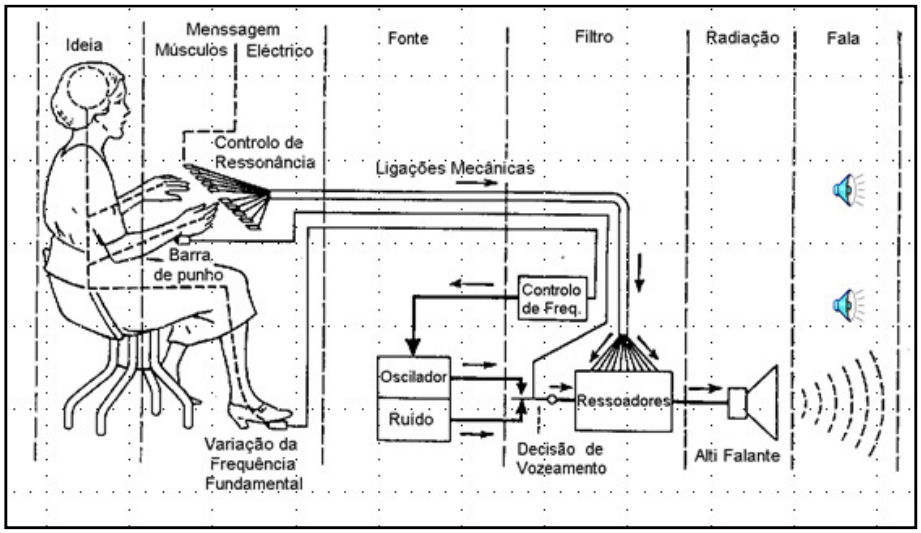

Figura 8: VODER. (Fonte: ptolemy.berkeley.edu)

Já em 1950, Frank Cooper desenvolveu um sintetizador da fala de um tipo diferente, funcionava como um inversor de um espectograma de som. Uma lâmpada produzia um raio de luz que era dirigida radialmente contra um disco rotativo, com 50 faixas concêntricas, cuja transparência variava de modo a produzir 50 parciais com uma frequencia fundamental de 120 Hz. A lus era posteriormente projetada contra um espectrograma cujo reflexo ou transparência correspondia ao nível de pressão sonora de cada parcial e dirigida a uma célula fotovoltaica para a obtenção das pressões sonoras. O som produzido era monótono.

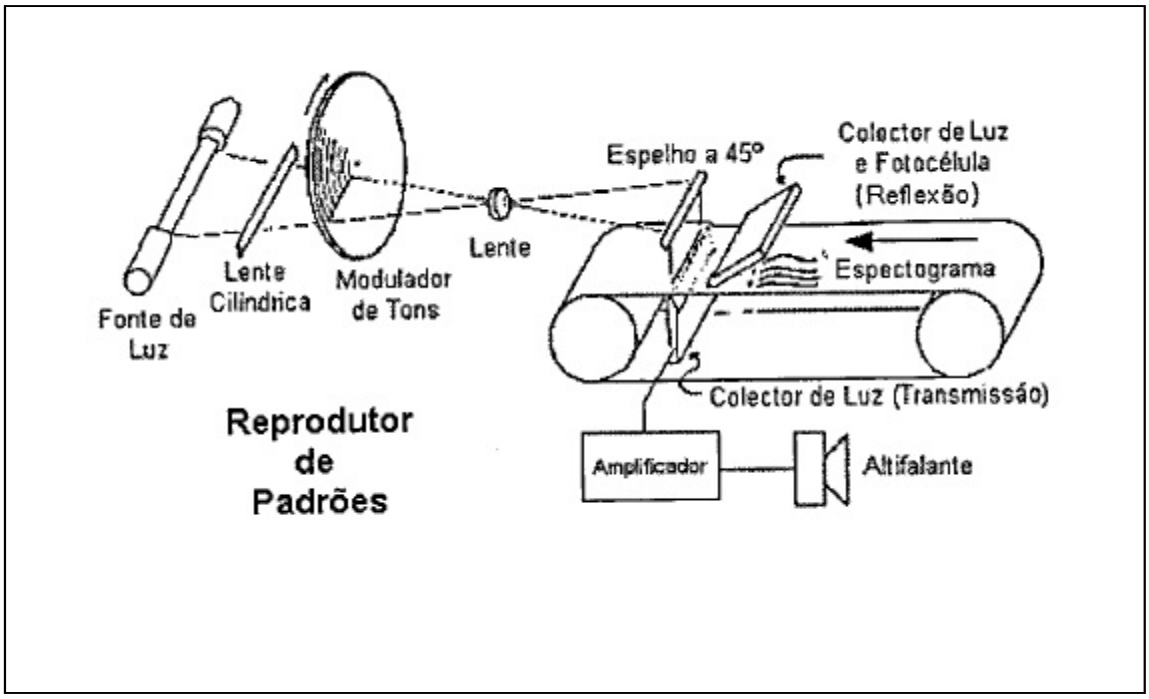

Figura 9: Esquema do Sintetizador de Fala de Frank Cooper. (Fonte: Barros, Maria J. A. Estudo Comparativo e Técnicas de Geração de Sinal para a Síntese da Fala. Porto: Faculdade de Engenharia da Universidade do Porto. 2002. Tese (Mestrado).

Em 1958, Rosen desenvolveu o primeiro circuito para a realização de síntese articulatória de forma automática, o *Dynamic Analog of the Vocal Tract*, ou DAVO (Klatt 1987).

Nos modelos desenvolvidos desde os anos 50, sinais elétricos passavam por filtros, que simulavam as propriedades ressonantes do trato vocal. A fonte de sinal era de um tom harmônico, para os sinais vozeados e ruído aperiódico para os segmentos não vozeados.

Foram tentadas duas diferentes aproximações:

- Simulação da articulação usando um elevado número de circuitos em cascata, cada um representando uma seção do trato vocal.
- Usando circuitos ressonantes que simulavam os formantes.

Um exemplo de uma implementação desta última aproximação é o OVE (*Orator Verbis Electris*), um sintetizador de formantes, de vogais, desenvolvido por Gunnar Fant, em 1953. Foi o primeiro sintetizador de formantes em cascata. [13]

A partir do final da década de 60, o desenvolvimento da síntese de voz ficou associado à tecnologia informática. Durante muitos anos, idéias sobre intensidade e força de articulação foram desacreditadas. Embora tivessem aparecido idéias como proeminência relativa, por exemplo, por Chomsky e Halle, foram consideradas como não tendo nenhuma base empírica.

Até o final dos anos 80 foram utilizadas as técnicas de primeira geração, ou seja, a síntese de voz por formantes e síntese articulatória.

A idéia de concatenar segmentos de fala foi realizada primeiro usando fonemas, mas dados os problemas de perda de qualidade devido aos efeitos de co-articulação inerentes a um contexto, optou-se por diferentes unidades, sendo a mais usada o difone (segmento que compreende desde a segunda metade de um fonema até a primeira metade do fonema seguinte).

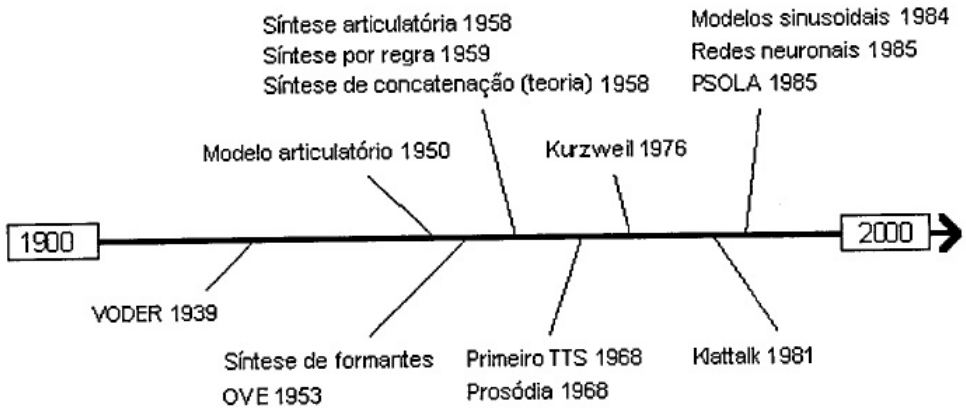

Figura 10: Referência cronológica de alguns avanços na síntese de voz. (Fonte: ptolemy.berkeley.edu)

Hoje, a principal preocupação na produção artificial de voz é a sua naturalidade. As duas características que descrevem a qualidade de um sintetizador de voz são a naturalidade *e* inteligibilidade. A naturalidade de um sintetizador de voz refere-se a até que ponto soa como a voz de uma pessoa real. A inteligibilidade de um sintetizador refere-se a facilidade da saída de poder ser entendida. O sintetizador ideal deve ser ao mesmo tempo natural e inteligível, e a cada tecnologia tenta conseguir o máximo de ambas. Algumas das tecnologias são melhores em naturalidade ou em inteligibilidade e as metas da síntese determinam com freqüência que aproximação deve seguir. Há duas tecnologias principais usadas para gerar fala sintética: Síntese concatenativa e síntese de formantes.

2.3.1. Síntese Concatenativa

A síntese concatenativa baseia-se na concatenação de segmentos de voz gravados. Geralmente, a síntese concatenativa produz os resultados mais naturais. No entanto, a variação natural da fala e as técnicas automatizadas de segmentação de formas de onda resultam em defeitos audiveis, que implicam uma perda de naturalidade.

Há três tipos básicos de síntese concatenativa.

2.3.1.1. Síntese por seleção de unidades

A síntese por seleção de unidades utiliza uma base de dados de voz gravada (mais de uma hora de fala gravada). Durante a criação da base de dados, a fala se segmenta em algumas ou todas das seguintes unidades: fonemas, sílabas, palavras, frases e orações. Normalmente, a divisão em segmentos realiza-se usando um reconhecedor de voz modificado para forçar seu alinhamento com um texto conhecido. Depois corrige-se manualmente, usando representações como a forma de onda e o espectrograma. Cria-se um índice das unidades na base de dados baseada em parâmetros acústicos da segmentação como a frequência fundamental, o pitch, a duração, a posição na sílaba e os fonemas vizinhos. Na execução, é feita a seleção de unidades. Este processo consegue-se tipicamente usando uma árvore de decisão especialmente ponderado.

A selecção de unidades dá a máxima naturalidade devido ao fato de não se aplicar muito processamento digital de sinais à fala gravada, o que com freqüência faz que o som gravado soe menos natural, ainda que alguns sistemas usam um pouco de processamento de sinal na concatenação para suavizar as formas de onda. Por exemplo, um sistema de síntese de voz para dar informações de voos pode ganhar em naturalidade se a base de dados foi construída a base gravações de informações de voos, pois será mais provável que apareçam unidades apropriadas e inclusive correntes inteiras na base de dados. No entanto, a máxima naturalidade com freqüência requer que a base de dados seja muito ampla, chegando em alguns sistemas aos gigabytes de dados gravados.

2.3.1.2. Síntese de difonos.

A síntese de difonos usa uma base de dados mínima contendo todos os difonos que podem aparecer em uma linguagem dada. O número de difonos depende do idioma : o espanhol tem uns 800 difonos, o alemão uns 2500. Na síntese de difonos, a base de dados contém um só exemplo da cada difono. Em tempo de execução, a prosodia de uma oração se sobrepõem a estas unidades mínimas mediante processamento digital de sinais, como codificação linear preditiva, PSOLA ou MBROLA.

A qualidade da fala resultante é geralmente pior que a obtida mediante seleção de unidades mas mais natural que a obtida mediante sintetizacção de formantes.

2.3.1.3. Síntese específica para um domínio

A síntese específica para um domínio concatena palavras e frases gravadas para criar saídas completas. Usa-se em aplicações onde a variedade de textos que o sistema pode produzir está limitada a um particular domínio, como anúncios de saídas de voos ou informação meteorológica. Esta tecnologia é muito singela de implementar, e usou-se comercialmente durante um longo tempo: é a tecnologia usada por aparelhos como relógios e calculadoras falantes. A naturalidade destes sistemas pode ser muito grande, porque a variedade de orações está limitada. No entanto, ao estar limitados a algumas frases e palavras da base de dados, não são de propósito geral e só podem sintetizar algumas combinações de palavras.

Em muitos sistemas, como por exemplo, nos sistemas de síntese de fala a partir de texto, é necessário produzir fala a partir de uma sequência fonética e respectiva informação (prosódica, energia, duração e frequencia fundamental), existindo um primeiro módulo que converte a sequencia ortográfica nestes parâmetros.Uma das aproximações no desenvolvimento destes sistemas é o armazenamento da forma de onda de cada segmento fonético, mas as dificuldades na alteração da informação prosódica e na modelação do efeito da coarticulação torna estes sistemas de dificil implementação.

2.3.2. Síntese de formantes

A síntese de formantes não usa amostras de fala humana na produção de voz artificial. Em lugar disso, a saída se cria usando um modelo acústico. Parâmetros como a frequência fundamental e os níveis de ruído variam durante o tempo para criar uma forma de onda ou fala artificial. Este método é conhecido também como síntese baseada em regras mas alguns alegam que muitos sistemas concatenativos usam componentes baseados em regras para algumas partes de seus sistemas, como o front-end, de modo que o termo não é suficientemente específico. Muitos sistemas baseados em síntese de formantes geram fala robótica e de aparência artificial, e a saída nunca poder-se-ia confundir com a voz humana. No entanto, a naturalidade máxima não é sempre a meta de um sintetizador de voz, e estes sistemas têm algumas vantagens sobre os sistemas concatenativos.

A síntese de formantes pode ser muito inteligivel, inclusive a altas velocidades, evitando os defeitos acústicos que podem aparecer com frequência nos sistemas concatenativos. A síntese de voz de alta velocidade é com freqüência usada pelos deficientes visuais para utilizar computadores com rapidez. Por outra lado, os sintetizadores de formantes são com freqüência programas mais pequenos que os sistemas concatenativos porque não precisam de uma base de dados de amostras de voz gravada. Desta forma não exigem sistemas com grande memória e a capacidade de processamento. Por último, dado que os sistemas baseados em formantes têm um controle total sobre todos os aspectos da fala produzida, podem incorporar uma ampla variedade de tipos de entonações.

2.3.3. Outros métodos de síntese

A síntese articulatória tem sido um método de interesse puramente acadêmico até pouco tempo. Baseia-se em modelos computacionais do trato vocal e o processo de articulação. Poucos modelos são suficientemente avançados ou eficientes computacionalmente para ser usados em sistemas comerciais de síntese de voz. Uma exceção notável é o sistema baseado em NeXT, originalmente desenvolvido e comercializado por *Trillium Sound Research Inc*, que passou mais tarde a ter uma licença GPL e se continuou como *gnuspeech*, sendo um projeto GNU. O sistema, que foi comercializado pela primeira vez em 1994, proporciona uma conversão texto - voz articulatoria completa mediante uma analogia de guia de onda ou linha de transmissão

dos tratos vocal e nasal humanos, controlados pelos Modelos de Região Distintiva de Carré que está baseado no trabalho de Gunnar Fant e outros do laboratório *Stockholm Specch Technology Lab do Royal Institute of Technology* sobre a análise da sensibilidade de formantes. Este trabalho mostrou que os formantes em um tubo ressonante podem ser controlados por oito parâmetros que correspondem aos articuladores disponíveis no trato vocal humano natural.

A síntese híbrida reune os aspectos das sínteses concatenativa e de formantes para minimizar os defeitos acústicos quando é feita a concatenação dos segmentos.

A síntese baseada em HMM é um método de síntese baseado em Modelo oculto de Márkov,Modelos ocultos de Márkov, (HMM em inglês). Neste sistema, a fala espectro de frequências (trato vocal), frequência fundamental (fonte vocal), e a duração (prosodia) são modeladas simultaneamente por modelos ocultos de Márkov.

2.4. Processamento de sinais

O termo sinal geralmente se refere a uma função de uma ou mais variáveis que contém informações sobre o comportamento ou natureza de alguns fenômenos. Um sistema pode ser definido como um dispositivo físico que realiza operações sobre o sinal. Quando o sinal é passado em um sistema diz-se que o sinal foi processado. Um dado sistema é caracterizado pelo tipo de operação que ele realiza sobre o sinal e estas operações são denominadas de processamento de sinais. É conveniente ressaltar que um sistema não inclui apenas dispositivos físicos (*hardware*), mas também operações realizadas por procedimento algorítmico (*software*).

Uma das primeiras tarefas da análise de sinais é entender quais informações podem ser obtidas de um dado sinal e como extrair estas informações.

A maioria dos sinais encontrados na natureza é analógica, ou seja, são contínuos no tempo. Porém, processar um sinal na forma analógica envolve uma série de fatores relacionados a limitações físicas de reconfiguração de *hardware* e controle da precisão do dispositivo de processamento. No entanto, um sinal digital pode ser fácil e rapidamente re-configurado, bastando, alterações em um algoritmo computacional. Sinais digitais podem ser facilmente armazenados e permitem o uso de algoritmos mais sofisticados no processamento. Além disso, o processamento digital pode ter custo muito menor que o processamento na forma analógica.

A principal vantagem do processamento analógico sobre o digital é a resposta precisa, e em tempo real, aos estímulos recebidos, através dos circuitos eletrônicos analógicos, devido à ausência de quantização de amplitude e discretização no tempo.

Por outro lado, o processamento digital possui três vantagens principais: imunidade, pois não existe a influencia de agentes que distorcem os valores de componentes passivos nos circuitos eletrônicos, tais como resistores e capacitores, flexibilidade, sendo todo o processamento via *software* e repetitividade.

2.5. Filtros digitais

Filtros digitais são filtros que atuam sobre sinais representados na forma digital, são elementos que irão exercer certo processamento sobre os dados do sinal, produzindo outro sinal, no qual terá uma modificação da informação contida no sinal original. Filtros podem ser usados, por exemplo, para segregar informações de faixas de freqüências diferentes, para remoção de ruídos, ou ainda para ressaltar determinadas informações.

A utilização de filtros digitais ao invés dos analógicos traz algumas vantagens. Os filtros digitais são programáveis, ou seja, o seu funcionamento é determinado por um programa armazenado em uma memória. Mudar esse programa é extremamente fácil, e não é necessário redesenhar o circuito do filtro, como é o caso dos filtros analógicos. Os filtros digitais são facilmente implementados e testados, já que não é necessário a construção de circuitos especiais para cada implementação, como é o caso dos filtros analógicos.

O filtro digital nada mais é que um sistema que realiza uma combinação linear de um sinal de entrada com certos coeficientes, para obter um sinal de saída com determinadas características de freqüência selecionadas. Os parâmetros mais relevantes de um filtro digital são:

- Freqüência de corte: é a freqüência para a qual o filtro já passa a ter uma atenuação maior ou igual a -3dB (aproximadamente 70,7%), que é o ponto onde termina a banda de passagem e inicia a banda de transição.
- Freqüência de rejeição: definida neste trabalho como a freqüência para a qual o filtro já passa a ter uma atenuação maior ou igual a 95% da

atenuação máxima, que é o ponto onde termina a banda de transição e se inicia a banda de rejeição.

- Banda de passagem: é a faixa de freqüência anterior a freqüência de corte.
- Banda de transição: é a faixa de freqüências que se inicia no final da banda de passagem e termina no início da banda de rejeição.
- Banda de rejeição: faixa de freqüências posterior à freqüência de rejeição.
- Máxima variação de ganho na banda de passagem.
- Mínima atenuação na banda de rejeição.
- Tipo:

  - Resposta ao impulso finita (*finite impulse response* – FIR): quando a quantidade de coeficientes do filtro digital, no domínio do tempo, é finita.

  - Resposta ao impulso infinita (infinite impulse response –IIR): quando a quantidade de coeficientes do filtro digital, no domínio do tempo, é infinita.
- Função: passa-baixa, passa - alta, passa-faixa e rejeita-faixa.
- Ordem: número de pólos da função de transferência do filtro. Um filtro digital com N+1 coeficientes possui ordem N. À medida que a ordem do filtro aumenta, sua resposta em freqüência fica mais próxima da ideal (a banda de transição é a mais estreita).
- Fase: linear (atraso constante da saída para toda a faixa de freqüência) ou não linear.

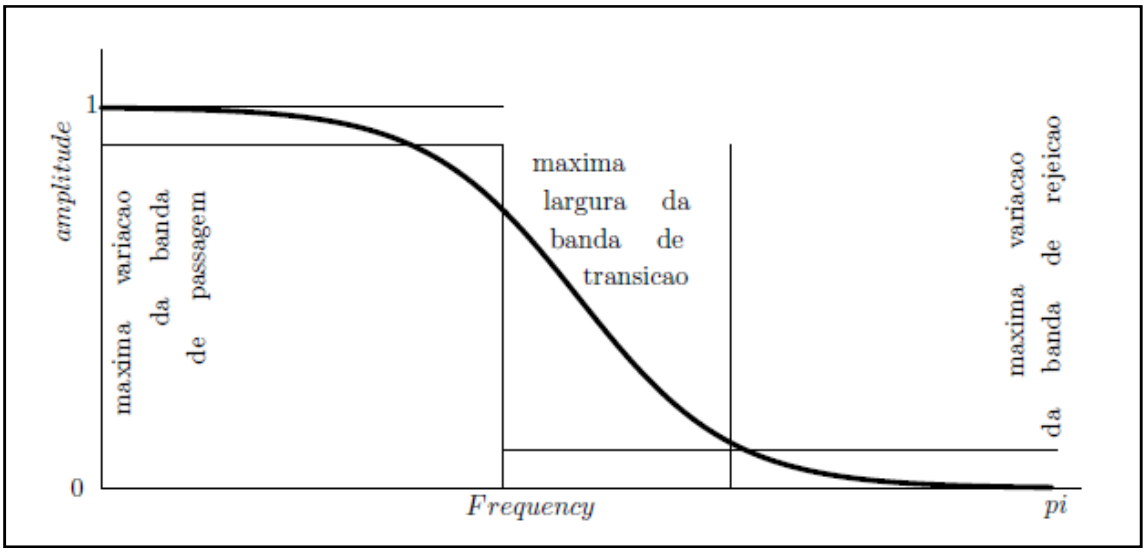

Figura 11: Principais parâmetros de um filtro digital. (Fonte: Vieira, Lucimar S. Conversão de Voz Baseada na Transformada Wavelet. São Carlos: IFSC –USP,2007. Dissertação (Mestrado).

A unidade de medida dB utilizada acima, chamada decibel, corresponde a uma unidade logarítmica utilizada para facilitar a mensuração de alguns parâmetros dos filtros digitais: dB = 20 log.

2.6. Teorema da Convolução

Este teorema enuncia que a multiplicação de dois sinais discretos no domínio da freqüência, H[z] e X[z], corresponde à convolução dos mesmos no domínio do tempo, h[n] e x[n]. A convolução, y[ ], dos dois sinais discretos x[ ] e h[ ], representada pelo símbolo *, é dada por:

$$y[] = x[] * h[] = \sum_{k=0}^{M-1} h_k x_{n-k} \quad (1)$$

Onde *M* é o número de amostras de *h*.

2.7. Resposta ao impulso de um filtro digital

A resposta ao impulso é a resposta do filtro para uma entrada impulsiva $\delta[\ ]$ = {1,0,0,0,...0}, que corresponde aos coeficientes do filtro digital no domínio do tempo, ou seja, a Transformada de Fourier inversa da função de transferência do filtro.

2.8. Amostragem de sinais contínuos

Os sinais de tempo – discreto ou digitais estão definidos apenas para alguns instantes de tempo específicos; estes instantes de tempo não precisam ser eqüidistantes, mas, para garantir a simetria temporal do sinal, normalmente são escolhidos desta forma.

Um sinal digital pode ser obtido através da amostragem de um sinal analógico. Existem muitas maneiras de se amostrar um sinal, mas a mais usada é a uniforme ou periódica, ou seja, realizada em intervalos de tempo eqüidistantes, T.

Seja $x_{a\,(t)}$ um sinal analógico, para efeito de simplificação, nas deduções necessárias, serão utilizadas oscilações harmônicas de amplitude (A), fase (θ) e freqüência (F). Assim, pode-se considerar $x_{a\,(t)}$, como:

$$x_a(t) = Acos(2\pi Ft + \theta) = Acos\,(\Omega t + \theta) \tag{2}$$

$$\Omega = 2\pi F \tag{3}$$

Define-se

$$x(n) = x_a(nT), -\infty < n < \infty \tag{4}$$

Em que n é um número inteiro, x(n) é o sinal discreto obtido ao se tomas amostras de $x_a(t)$ a cada T segundos. T é chamado de *período de amostragem* ou *intervalo de amostragem* e, 1/T = Fs é chamada de freqüência de amostragem. Assim, x(n) é dado por:

$$x(n) = Acos(2\pi fn + \theta) = Acos(\omega n + \theta)$$

$$\omega = 2\pi f \tag{5}$$

Em que *f* é a freqüência do sinal digital.

A amostragem periódica estabelece relações entre a variável continua *t* e a variável discreta *n*, ou seja:

$$t = nT = \frac{n}{Fs} \tag{6}$$

Como conseqüência da equação (6) existe também uma relação entre a variável F (ou Ω = 2πF) para sinais analógicos e a variável f ou (ω = 2πf) para sinais digitais. Para estabelcer esta relação, considera-se o sinal analógico.

$$xa(t) = Acos(2\pi Ft + \theta) \tag{7}$$

Que amostrado periodicamente a Fs = 1/T fornece:

$$x_a(nT) \equiv x(n) = Acos(2\pi FnT + \theta = Acos(\frac{2\pi nF}{Fs} + \theta) \tag{8}$$

Comparando a equação (8) com a equação (5), tem-se:

$$f = \frac{F}{Fs} \tag{9}$$

Ou

$$\omega = \Omega T \tag{10}$$

Pode-se também verificar, a partir da equação (1), que

$$-\infty < F < \infty$$

$$-\infty < \Omega < \infty \tag{11}$$

E, de maneira análoga, pode-se verificar, a partir da equação (5), que

$$\frac{-1}{2} < f < \frac{1}{2}$$

$$-\pi < \omega < \pi \tag{12}$$

Com o auxílio da equação (9), a inequação (12) pode ser reescrita como:

$$\frac{-1}{2} < \frac{F}{Fs} < \frac{1}{2} \tag{13}$$

Uma vez que, Fs=1/T a inequação (13) pode ser reescrita na forma

$$\frac{-1}{2T} = -\frac{Fs}{2} \leq F \leq \frac{Fs}{2} = \frac{1}{2T} \tag{14}$$

Que é equivalente a:

$$-\frac{\pi}{T} = -\pi Fs \leq \Omega \leq \pi Fs = \frac{\pi}{T} \tag{15}$$

Conclui-se, então que a amostragem periódica de um sinal analógico implica o mapeamento do intervalo infinito de freqüências F em um intervalo finito das freqüências f. O que resulta em:

$$Fmax = \frac{Fs}{2} = \frac{1}{2T}$$

$$\Omega max = \pi Fs = \frac{\pi}{T} \tag{16}$$

2.9. Teorema da Amostragem

Um dos problemas da amostragem de sinais é, dado um determinado sinal analógico, como escolher a freqüência de amostragem Fs. Para que isto seja feito corretamente, é necessário que se tenha informações sobre as freqüências presentes no sinal a ser analisado, mais precisamente, é necessário conhecer a máxima freqüência presente no sinal (*Fmáx*), uma vez que deve-se ter:

$$Fs > 2Fmax \tag{17}$$

Em que *Fmax* é a maior componente em freqüência presente no sinal analógico. Com *Fs* selecionada desta maneira, qualquer freqüência *Fi* presente no sinal analógico é mapeada em uma senóide de tempo discreto com freqüências dadas por:

$$\frac{-1}{2} \leq f_i = \frac{Fi}{Fs} \leq \frac{1}{2} \tag{18}$$

Se *Fs* for escolhida conforme a equação (18) todas as componentes em freqüência presentes no sinal analógico são representadas sem ambigüidade no sinal digital e a reconstrução do sinal pode ser feita sem distorções, através do uso de uma função de interpolação adequada.

A freqüência $Fn \leq 2Fmax$ é também conhecida como freqüência de Nyquist.

2.9.1. Aliasing

Aliasing corresponde às freqüências ‘ fantasmas’ que surgem em um sinal digitalizado, quando o correspondente analógico é amostrado a uma taxa insuficiente.

3. Descrição das atividades desenvolvidas

A primeira etapa do projeto consistiu na busca de algoritmos existentes para síntese de voz. Em seguida foi feita a pesquisa para compreender o processo natural de geração de voz pelo ser humano (fisiologia da fala). Através da revisão bibliográfica foi possível adquirir o conhecimento e as informações necessárias para o desenvolvimento do projeto.

Neste capítulo são apresentadas as fases do projeto e as atividades desenvolvidas em cada uma delas.

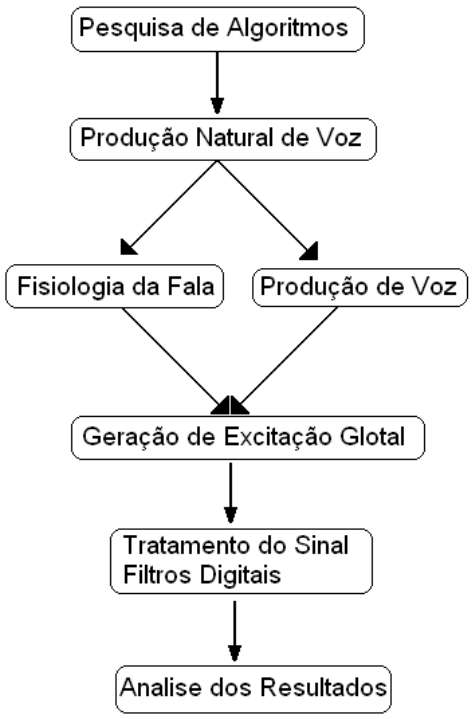

Figura 12: Fases do Projeto.

3.1. Modelagem da Fonte

Para elaborar o algoritmo para síntese de voz foi adotado o modelo fonte-filtro de produção de som, apresentado no capítulo 2, seção 32. A síntese de voz escolhida foi a síntese de formantes.

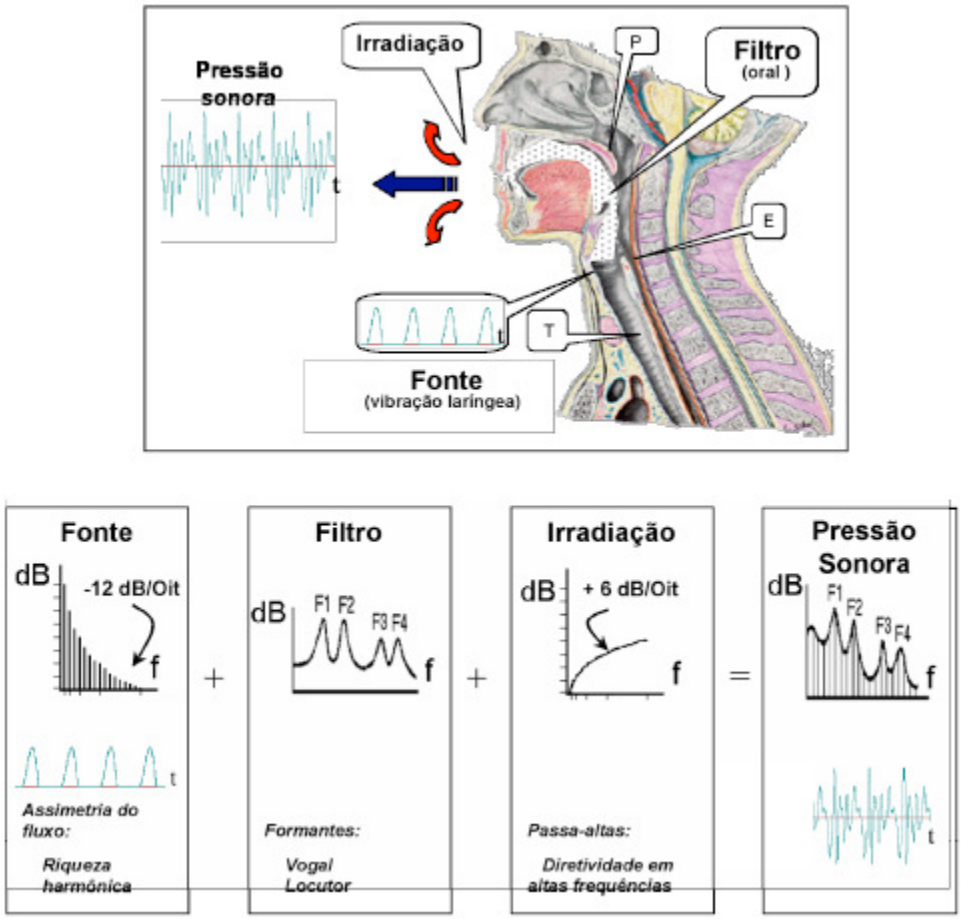

Figura 13: Esquemático do Processo de Geração de Voz.

A primeira etapa da elaboração do programa consistiu na modelagem da fonte (representando a excitação glotal – pulmão).

Modelagem da fonte de excitação:

$$g[n] = \frac{1}{2}(1 - \cos\left(\frac{\pi n}{N1}\right) \qquad para\ 0 \leq n \leq N_1$$

$$g[n] = \cos\left(\pi\frac{(n - N1)}{2N_2}\right) \qquad para\ N_1 \leq n \leq N_1 + N_2$$

$$g[n] = 0 \qquad caso\ contrário. \quad (19)$$

Sendo que N_1 corresponde a 10% das amostras e N_2 corresponde a 65% das amostras utilizadas.

Como o foco deste estudo foi a produção de sinais periódicos que produzem *voiced speech*, basicamente vogais, foram escolhidos diferentes períodos de *pitch,*

conhecido como F0. Para cada valor de F0, foi determinada a taxa de amostragem, quantidade de amostras necessárias, e calculados os valores de N_1 e N_2.

Após realizar a modelagem, foi implementada uma função em linguagem C para geração do sinal. O código é apresentado no anexo 5.1.

Utilizando o *software* Audacity®, um programa livre e gratuito, para edição de áudio digital, um arquivo branco do tipo *wave* com duração de cinco segundos foi gerado.

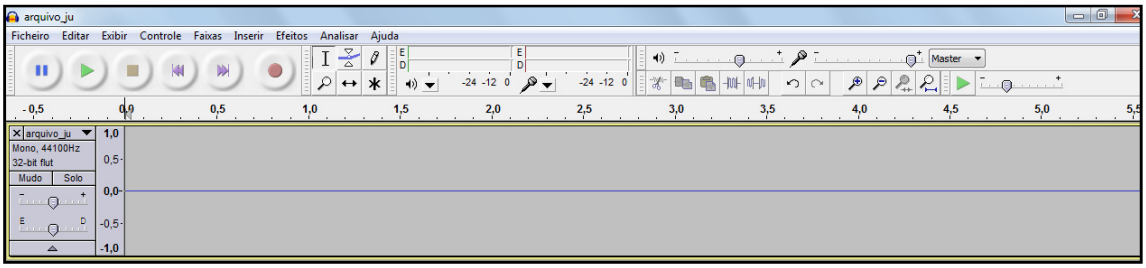

Figura 14: Arquivo wave "branco".

Fornecendo como entrada o arquivo wave “branco” ao algoritmo de geração de excitação e escolhendo a freqüência fundamental (*pitch*), aproximadamente 100 Hz, obteve-se como saída uma onda sonora pura de excitação.

Esse sinal temporal possui espectro repleto de harmônicas. Quando ele atravessa o trato vocal/nasal, o espectro é equalizado, ou modificado, de forma a ficar similar com a figura 17.

As freqüências de ressonância desse equalizador são chamadas formantes, abreviadas por F1, F2, ..., Fn.

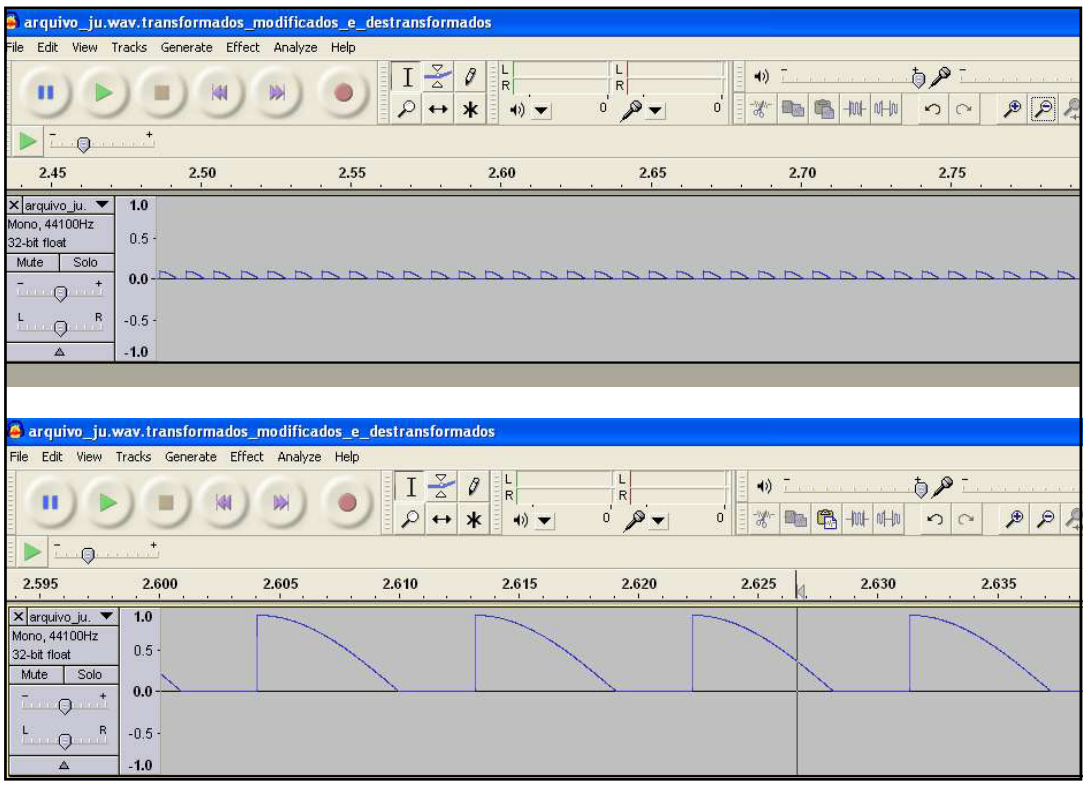

Figura 15: Sinal de excitção glotal obtido.

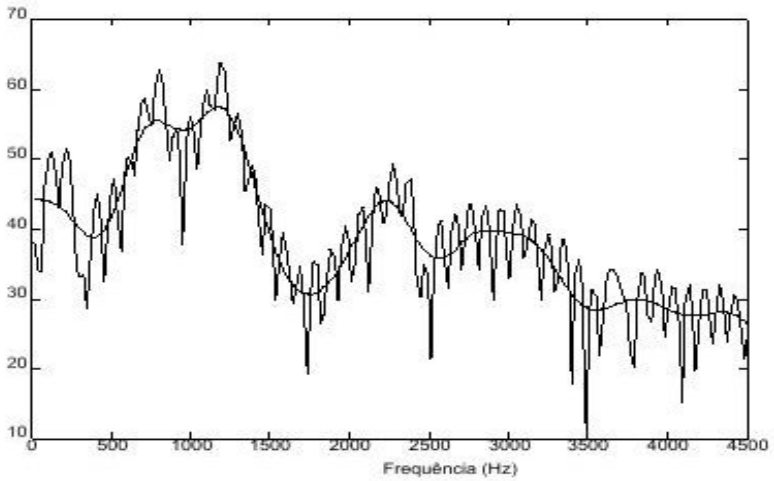

Figura 16: Espectro do sinal.

3.2. Tratamento do sinal – Filtro Digital

Para simular o trato vocal, foram desenvolvidos filtros digitais para equalizar o sinal. O primeiro filtro projetado foi o filtro passa-baixa IIR, aproximação de Bessel.

- As características da resposta passa-baixa de Bessel são:
- Declive na zona de transição muito menor do que num filtro Butterworth;

- Banda passante plana;
- Banda de rejeição sem ondulações;

A aproximação de Bessel é utilizada para produzir um defasamento linear da freqüência, comprometendo o declive.

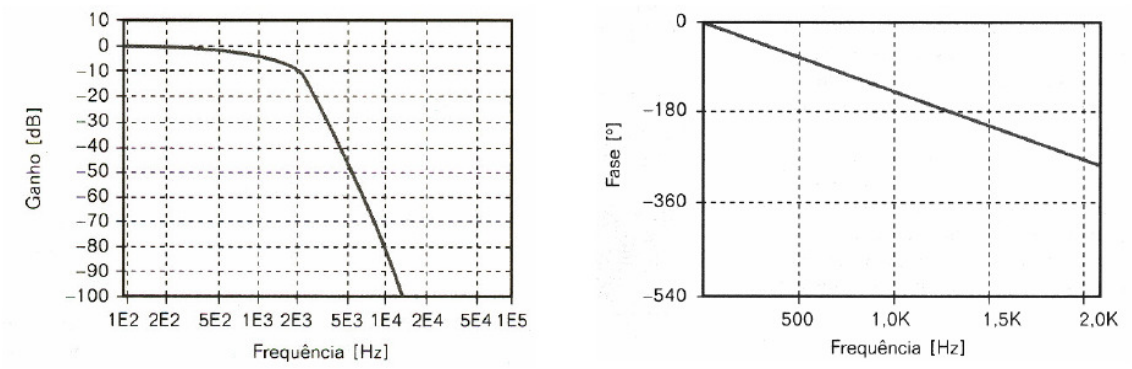

Figura 17: Aproximação Bessel.

O defasamento linear faz com que a distorção de um sinal seja mínima.

Foi implementado em C, um programa pra geração do filtro passa-baixa, apresentado no anexo 5.2.

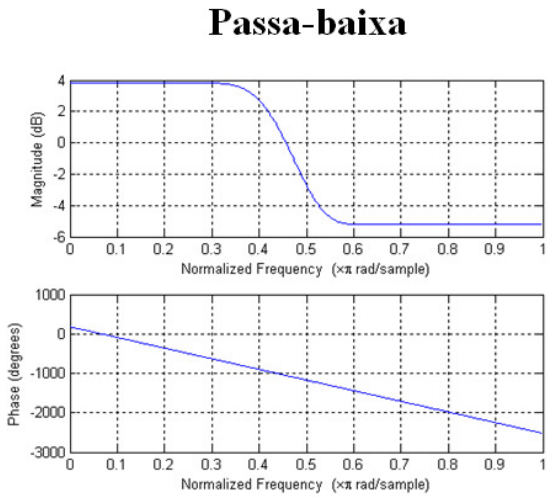

Figura 18: Filtros Passa-Baixa obtido.

É possível derivar o filtro passa-alta a partir do filtro passa-baixa. Isto é feito carregando os coeficientes do filtro passa-baixa em um vetor v1, outro vetor semelhante ao primeiro é criado v2, será o vetor dos coeficientes do filtro passa-alta. Invertem-se os elementos do vetor v2, e troca-se o sinal dos elementos que estão nas posições impares do vetor v2.

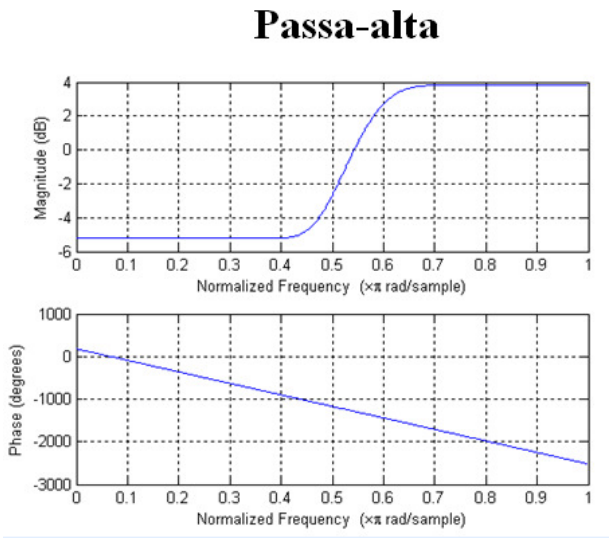

Figura 19: Filtro Passa - Alta obtida.

O Filtro Passa Faixa é obtido pela disposição em cascata (convolução) de um passa baixas e um passa altas.

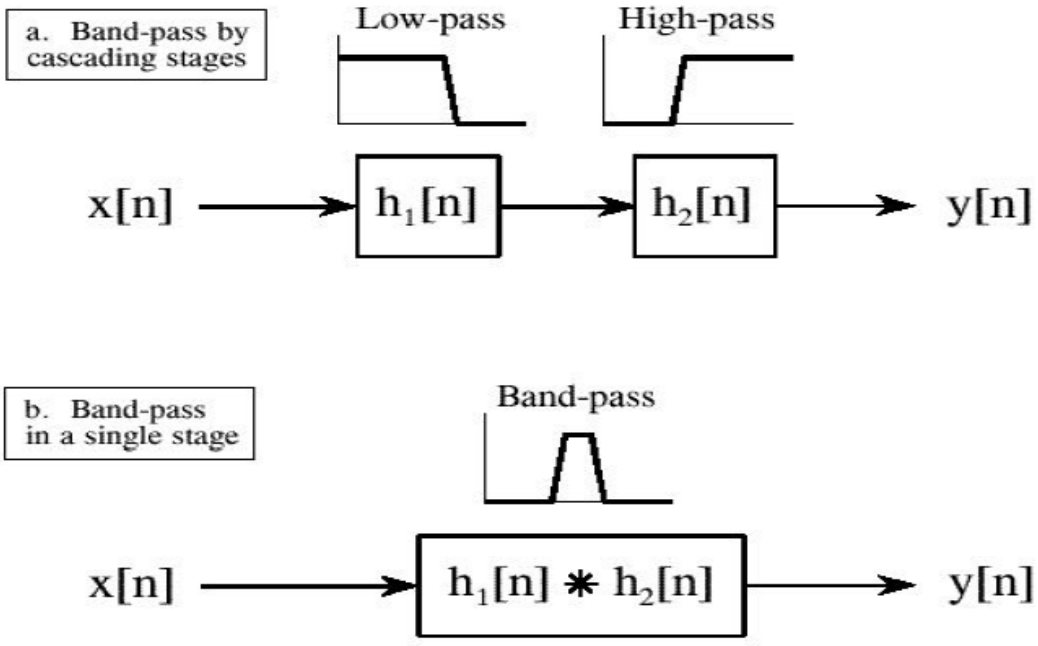

Figura 20: Método para obtenção do Filtro Passa-Faixa.

Os coeficientes de um dos filtros utilizados no projeto são apresentados na figura abaixo.

Passa-baixa		Passa-Alta		Passa-faixa
-0.0001170000		0.1727510000		-2.02119E-05
-0.0021870000		-0.2402500000		-0.000349697
-0.0074580000		0.2402500000		-0.00079106
-0.0078270000		-0.1727510000		-6.55525E-05
0.0169330000		0.0822690000		0.003382026
0.0822690000		-0.0169330000		0.009373898
0.1727510000		-0.0078270000		0.014922442
0.2402500000		0.0074580000		0.016338547
0.2402500000		-0.0021870000		0.012642656
0.1727510000		0.0001170000		0.006491871
				-0.012642656
				0.016338547
				-0.014922442
				0.009373898
				-0.003382026
				-6.55525E-05
				2.403816486

Figura 21: Coeficiente dos Filtros.

O filtro passa – faixa obtido é apresentado na figura 23.

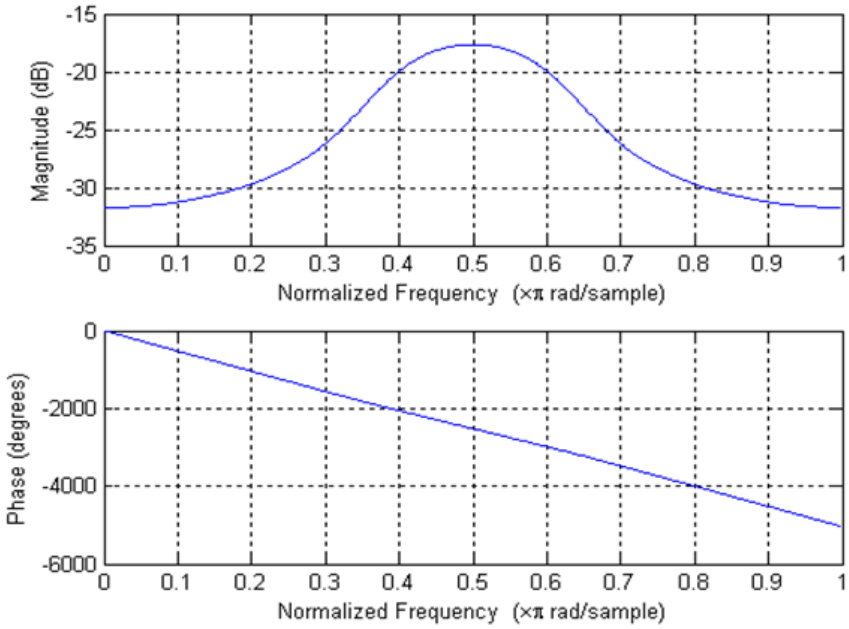

Figura 22: Filtro Passa-Faixa.

Ao terminar o projeto dos filtros, foi implementado em C, o algoritmo para realizar a filtragem do sinal de pura excitação. O algoritmo é apresentado no anexo 5.3.

3.3. Sinal Filtrado

Foi feita a avaliação nos domínios do tempo e da freqüência dos sinais obtidos. Para isto foi utilizado o software MatLab. O sinal filtrado é apresentado a seguir:

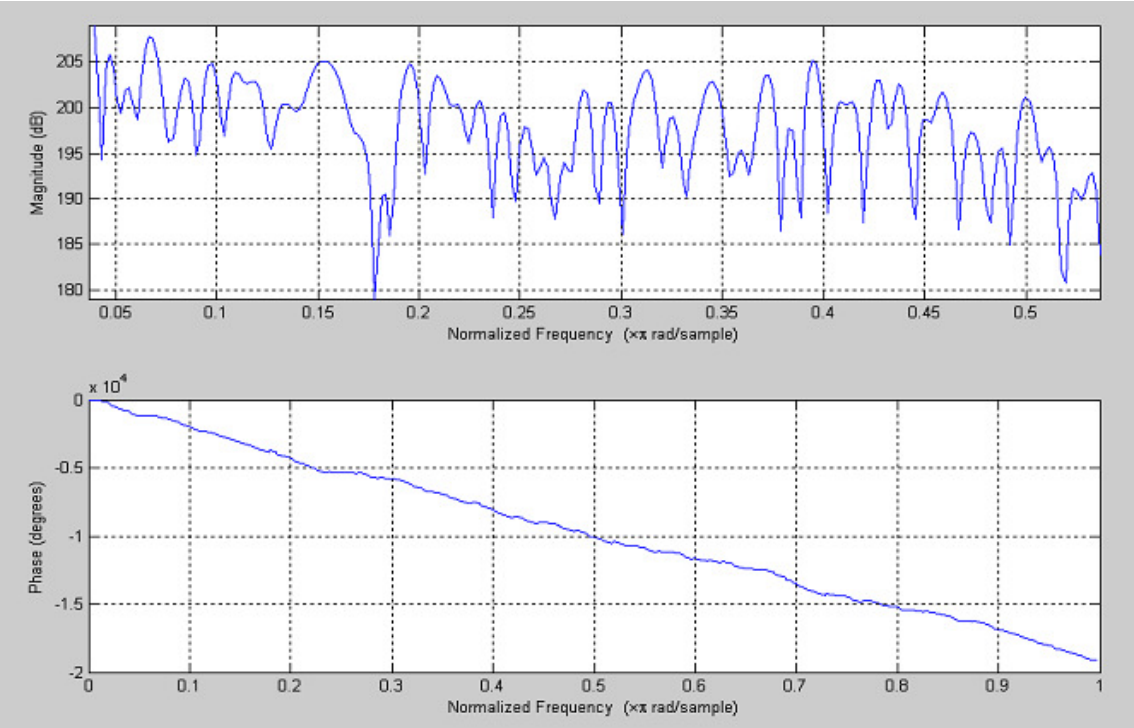

Figura 23: Sinal Filtrado.

Como foi utilizada a síntese de formantes, o algoritmo desenvolvido apresentou a vantagem de ser mais simples do que um algoritmo para síntese por concatenação e exigiu menor capacidade de processamento e de memória por não utilizar uma base de dados com amostras de vozes humanas.

Porém, o sinal gerado apresenta características de um sinal robótico, pouco parecido com a voz humana.

Os sinais produzidos são sinais periódicos, vogais.

Para verificar a eficiência do algoritmo, foi feita a síntese de uma vogal ‘a’. A comparação dos formantes do sinal gerado com um sinal de voz ‘a’ mostrou que ambos são muito próximos, comprovando a qualidade do algoritmo gerado.

4. Conclusões

Através da elaboração deste trabalho foi possível compreender o mecanismo básico de produção de voz humana. Após entender o processo de gerar de voz pelo ser humano, foi feito um estudo sobre a história da síntese de voz e quais as técnicas atuais utilizadas.

Obtida esta base, optou-se pela síntese de formantes, considerada mais rápida que a síntese concatenativa, porém o som produzido é mais metálico, robótico.

O modelo fonte-filtro utilizado possibilitou o desenvolvimento de um algoritmo capaz de produzir um sinal de excitação, que foi posteriormente equalizado com o uso de filtros digitais.

O resultado do projeto foi o desenvolvimento de um algoritmo simples, que não exige grande espaço de memória e processamento. Além disso, é possível que as freqüências do *pitch* e formantes sejam alteradas de acordo com o sinal final que deseja-se obter.

5. Trabalhos Futuros

Além disso, esta em andamento o projeto de desenvolvido do *hardware* de um transmissor AM, onde podem ser captados os sinais de voz sintetizados e transmitidos por radio freqüência, e um receptor AM, no qual podem ser recebidos os sinais de voz transmitidos por radio freqüência e passados ao processador, além das fontes de alimentação para, da forma apropriada, suprir as tensões utilizadas pelos dispositivos. O transmissor AM é basicamente um dispositivo que transmite um sinal com modulação em amplitude. Nesse caso, trata-se dos sinais de vozes captados que, em seguida, serão recebidos por radio freqüência pelo receptor AM, que faz a demodulação e os passará ao DSP / DSPic para tratamento e classificação, conforme o algoritmo embarcado. Esses dispositivos devem ser alimentados com uma fonte CC (corrente contínua), que é uma configuração de circuito que transforma, retifica e acondiciona a tensão de corrente alternada da rede elétrica, em tensão de corrente contínua. A implementação do sistema está planejada conforme ilustra a figura abaixo.

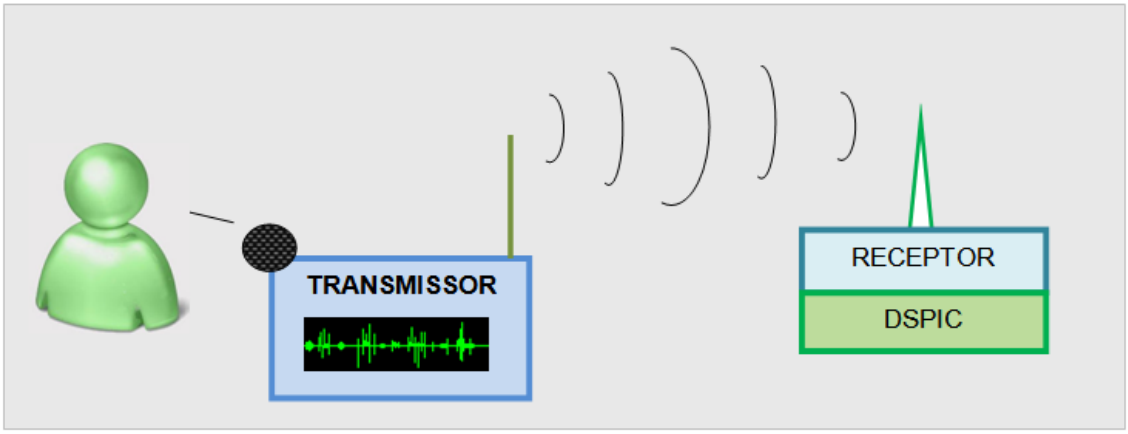

Figura 24: Arquitetura do Projeto Proposto.

O uso de transmissor e receptor AM justifica-se pelo privilégio de ter-se uma estação fixa, na qual serão tratados os dados, e uma estação móvel, com a qual poderá se captar os sinais de voz remotamente, e, com isso, utilizar o escopo do projeto dentro de um raio a partir da estação fixa, livremente, possibilitando maior número de aquisições, testes, e aplicações dentro das áreas de interesse citadas.

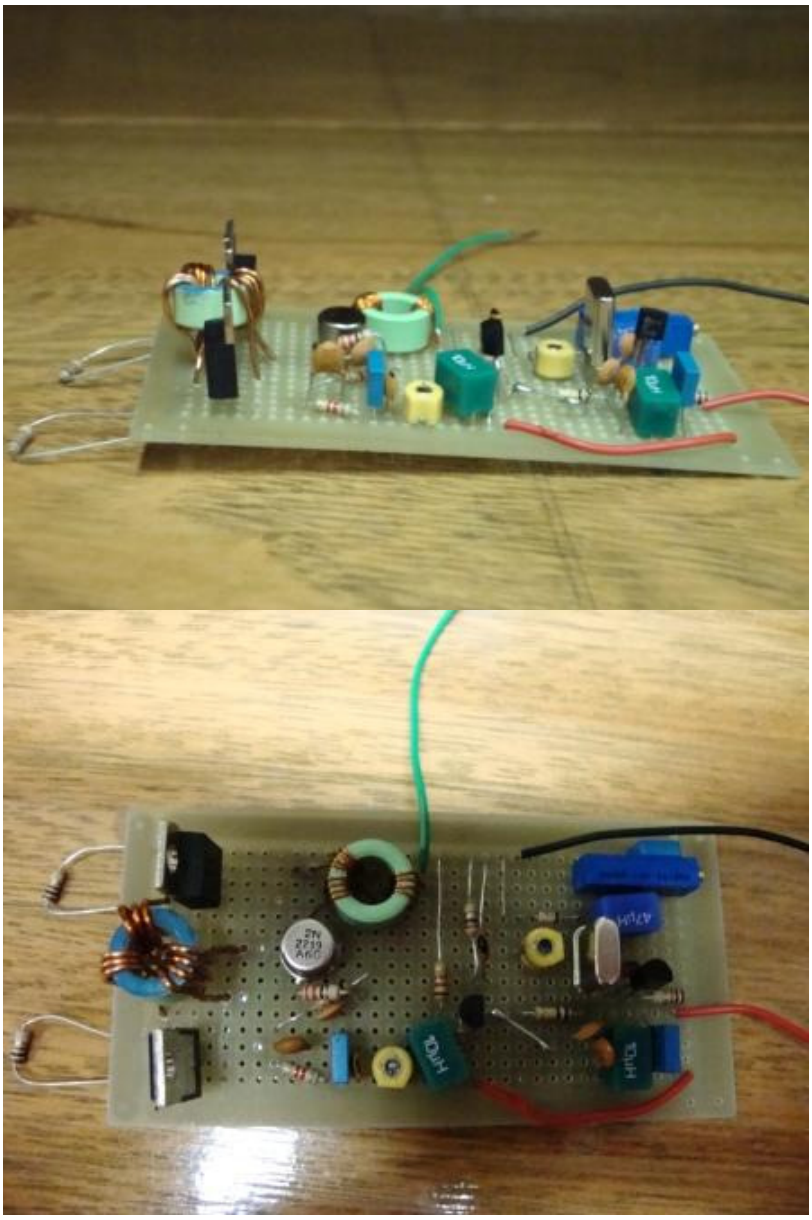

Figura 25: Transmissor AM.

6. Anexos

6.1. Algoritmo para geração do sinal de excitação

```
void modifica_dados_brutos(double* dados, long tamanho)
{   int fl,n;
    float N1,N2,N3;
    int amostras;
    amostras=400;
    N1=(0.1)*amostras;
    printf("N1=%f",N1);
    N2=(0.65)*amostras;
    printf("N2=%f",N2);
    N3=N1+N2;
    printf("N3=%f",N3);
    for(n=0;n<tamanho+1;n++)
    {fl=n;
    if(fl>amostras){fl=fl%amostras;} //se fl for maior que o numero de amostras, divide por amostras e pega o resto
    if(fl<N1)
    {dados[n]=1000*(1/2)*(1-cos(3.1415*fl/N1));
            }
    else {if(fl<N3){dados[n]=1000*cos(3.1415*(fl-N1)/(2*N2));}
        else{dados[n]=0;}  }
                }
                printf("\n\n");
                for(int j =8001;j<10000;j++)
                printf(" %ld,",dados[j]);
                getch();
```

6.2. Programa para geração do filtro passa-baixa

```
#include<iostream.h>

#include<stdio.h>

#include<math.h>

main()

{

double bessel(double);

const double pi=3.1415927;

double alpha=0.25*pi;//frequencia de corte, entre 0 e pi. No caso, 0.25*pi corresponde a 1/4 da maxima frequencia

double wp=0.4*3.1415927;

double ws=0.6*3.1415927;

double delta=0.01;

double a=-20*log10(delta);

int m=1+(int)((a-8)/(2.285*(ws-wp)));

double beta;

if(a>50)

        beta=0.1102*(a-8.7);

else if((a>=21)&&(a<=50))

        beta=0.5842*pow((a-21),0.4)+0.07886*(a-21);

else

        beta=0;

double *h = new double[m+1];

for(int i=0;i<=m;i++)

        {

        h[i]=((sin(alpha*(i-m/2.0)))/(pi*(i-m/2.0)))*(bessel(beta*(sqrt(1-pow((i-m/2.0)/(m/2.0),2))))/bessel(beta));

        printf("\nh[%d] = %f",i,h[i]);

        }

cout<<"\n\n\n";

}
```

```
//---------------------------------------------
double bessel(double k)
{
double fat(int);
double sum=1;
for(int i=1;i<=25;i++)
        sum+=pow(((1.0/fat(i))*(pow(k,i))),2.0);
return(sum);
}
//---------------------------------------------
double fat(int k)
{
if (k==1)
        return(1);
else
        return(k*(fat(k-1)));
}
```

6.3. Algoritmo para realizar a filtragem do sinal.

```
////////////////////////////////////////////////////////////////////////////////
/////////////////

double c[29]; //c e' o filtro

//preencher c[]

 c[0]= -0.000000;

 c[1]= -0.000000;

 c[2]= 0.000000;

 c[3]= -0.000000;

 c[4]= -0.000002;

 c[5]= -0.000000;
```

```
 c[6]= 0.000011;
 c[7]= 0.000000;
 c[8]= -0.000037;
 c[9]= 0.000000;
 c[10]= 0.000109;
 c[11]= 0.000000;
 c[12]= -0.000272;
 c[13]= -0.000000;
 c[14]= 0.000584;
 c[15]= -0.000000;
 c[16]= -0.000435;
 c[17]= -0.000000;
 c[18]= -0.001136;
 c[19]= 0.000000;
 c[20]= 0.005559;
 c[21]= -0.000000;
 c[22]= -0.013837;
 c[23]= -0.000000;
 c[24]= 0.025517;
 c[25]= 0.000000;
 c[26]= -0.038188;
 c[27]= 0.000000;
 c[28]= 0.048120;
 c[29]= -0.000000;
double s=0;
for(int i=0;i<(int)(sizeof(c)/sizeof(double));i++)
    s+=c[i];
for(int i=0;i<(int)(sizeof(c)/sizeof(double));i++)
    c[i]*=1.0/(s*s);
```

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

for(long
n=(int)(sizeof(c)/sizeof(double));n<tamanho+(int)(sizeof(c)/sizeof(double))-
1;n++)

   {

   y[n]=0;

   for(int k=0;k<(int)(sizeof(c)/sizeof(double));k++)

      y[n]+=dados[n-k]*c[k];

   }

for(long
i=(int)(sizeof(c)/sizeof(double));i<tamanho+(int)(sizeof(c)/sizeof(double))-1;i++)

   dados[i-(int)(sizeof(c)/sizeof(double))]=y[i];

 for(int y= 0;y<200;y++)

                {

                printf( "\n %ld",dados[y]);

                }              }
```

6.4. Código do programa utilizado no trabalho

```
//plus wavelets Prof Guido - Out-
2007

#include<iostream.h>

#include<stdio.h>

#include<math.h>

#include<string.h>

#include<conio.h>

#include<stdlib.h>

  //

//-------------------------------------

main(int i,char* n[])

{

void
modifica_dados_brutos(double*,long
);

short
converte2de8para1de16(unsigned
char, unsigned char);

void converte1de16para2de8(short,
unsigned char*, unsigned char*);

FILE* fr;

FILE* fw;

char arquivo_wav_resultante[200];

arquivo_wav_resultante[0]='\0';

char nome_arquivo[2000];

nome_arquivo[0]='\0';
```

```
cout<<"\nnome do arquivo :";

scanf("%s",&nome_arquivo);

//fgets(nome_arquivo,2000,stdin);

cout<<"O nome do arquivo digitado:"<<nome_arquivo; //nao usar espaços

strcat(nome_arquivo,":/Documents and Settings/TEMP.DPROEESC/Desktop/arquivo_ju.wav");

strcat(&arquivo_wav_resultante[0],nome_arquivo);

cout<<"\nArquivo wav e´:"<<arquivo_wav_resultante;

getch();

strcat(&arquivo_wav_resultante[0],".transformados_modificados_e_destransformados.wav");

cout<<"\nArquivo wav modificado fica:´"<<arquivo_wav_resultante;

//cout<<"\n\nPasso 2";

//getch();

if(((fr=fopen(nome_arquivo,"rb"))!=NULL)&&(((fw=fopen(arquivo_wav_resultante,"wb"))!=NULL)))

{

//cout<<"\n\nAnalisando 3";

//getch();

struct

    {

    unsigned char riff[4];

    unsigned long len;

    } riff_header;

fread(&riff_header,sizeof(riff_header),1,fr);

fwrite(&riff_header,sizeof(riff_header),1,fw);

cout<<"\nArquivo do tipo: "<<riff_header.riff[0]<<riff_header.riff[1]<<riff_header.riff[2]<<riff_header.riff[3];

cout<<"\nTamanho excluindo header:"<<riff_header.len;

////////////////////////////////////////////////////////

//cout<<"\n\nAnalisando 4";

//getch();

unsigned char wave[4];

fread(&wave,sizeof(wave),1,fr);

fwrite(&wave,sizeof(wave),1,fw);

//cout<<"\nSub-Tipo:"<<wave[0]<<wave[1]<<wave[2]<<wave[3];

////////////////////////////////////////////////////////

//cout<<"\n\nAnalisando 5";

//getch();

struct

    {

    unsigned char id[4];

    unsigned long len;

    } riff_chunk;


fread(&riff_chunk,sizeof(riff_chunk),1,fr);

fwrite(&riff_chunk,sizeof(riff_chunk),1,fw);

//cout<<"\nIdentificador:"<<riff_chunk.id[0]<<riff_chunk.id[1]<<riff_chunk.id[2]<<riff_chunk.id[3];
```

```
//cout<<"\nComprimento do chunk apos header:"<<riff_chunk.len;



////////////////////////////////////////////////////////////////////

//cout<<"\n\nAnalisando 6";

//getch();

struct

    {

    unsigned short formattag;

    unsigned short numberofchannels;

    unsigned long samplingrate;

    unsigned long avgbytespersecond;

    unsigned short blockalign;

    } wave_chunk;

fread(&wave_chunk,sizeof(wave_chunk),1,fr);

fwrite(&wave_chunk,sizeof(wave_chunk),1,fw);

//tratamento de uma excessao que costuma aparecer em alguns arquivos wav... Ocorreto seriam

// 16 bits, as vezes aparecem 18 ou mais...

//cout<<"\n\nPasso7";

//getch();

if(riff_chunk.len>16)

   {

//cout<<"\nTratamento de excesso";

//getch();

      unsigned char excesso;

   for(int i=0;i<riff_chunk.len-16;i++)

    {

fread(&excesso,sizeof(excesso),1,fr);

fwrite(&excesso,sizeof(excesso),1,fw);

    }

}

//fim do tratamento de excesso

//cout<<"\nCategoria do formato: "<<wave_chunk.formattag;

//cout<<"\nNumero de canais: "<<wave_chunk.numberofchannels;

//cout<<"\nTaxa de amostragem: "<<wave_chunk.samplingrate;

//cout<<"\nMedia do num. de bps: "<<wave_chunk.avgbytespersecond;

//cout<<"\nAlinhamento do bloco em bytes: "<<wave_chunk.blockalign;

//cout<<"\n\nAnalisando 8";

//getch();

////////////////////////////////////////////////////////////////////

if(wave_chunk.formattag==1)//FCM

  {

 //cout<<"\n\nNao e' comprimido";

//getch();

  int resolucao=(wave_chunk.avgbytespersecond * 8)/(wave_chunk.numberofchannels *
```

```
wave_chunk.samplingrate);//pq nao
bitssample

  //cout<<"\nResolucao:
"<<resolucao;

  struct

     {

     unsigned char data[4];

     unsigned long chunk_size;

     } header_data_chunk;

fread(&header_data_chunk,sizeof(h
eader_data_chunk),1,fr);

fwrite(&header_data_chunk,sizeof(h
eader_data_chunk),1,fw);

cout<<"\nIdentificacao:
"<<header_data_chunk.data[0]<<he
ader_data_chunk.data[1]<<header_
data_chunk.data[2]<<header_data_
chunk.data[3];

cout<<"\nTamanho do chunk de
dados:
"<<header_data_chunk.chunk_size;

cout<<"\nNumero de frames para
mostrar:
"<<header_data_chunk.chunk_size/
wave_chunk.blockalign;

cout<<"\n\nAnalisando 10";

getch();

long
tamanho_da_janela=header_data_c
hunk.chunk_size/wave_chunk.block
align;

cout<<"\nTamanho da janela:
"<<tamanho_da_janela;

if((resolucao==8)&&(wave_chunk.nu
mberofchannels==1))

 {

//cout<<"\n\n8 bits 1 canal";

//getch();

  unsigned char waveformdata;

  double* amostras_no_tempo =
new double[tamanho_da_janela];

  for(long
i=0;i<tamanho_da_janela;i++)

    {


fread(&waveformdata,sizeof(wavefo
rmdata),1,fr);


amostras_no_tempo[i]=(double)wav
eformdata;

     }


modifica_dados_brutos(&amostras_
no_tempo[0],tamanho_da_janela);

  for(long
i=0;i<tamanho_da_janela;i++)

   {

   waveformdata=(unsigned
char)amostras_no_tempo[i];


fwrite(&waveformdata,sizeof(wavefo
rmdata),1,fw);

   }

}

else
if((resolucao==8)&&(wave_chunk.nu
mberofchannels==2))

{

//cout<<"\n\n 8bits 2 canais";

//getch();

     unsigned char
waveformdata_right;
```

```
    unsigned char
waveformdata_left;

    double* amostras_no_tempo_left
= new double[tamanho_da_janela];

    double*
amostras_no_tempo_right = new
double[tamanho_da_janela];

    for(long
i=0;i<tamanho_da_janela;i++)

      {


fread(&waveformdata_left,sizeof(wa
veformdata_left),1,fr);


fread(&waveformdata_right,sizeof(w
aveformdata_right),1,fr);


amostras_no_tempo_right[i]=(doubl
e)waveformdata_right;


amostras_no_tempo_left[i]=(double)
waveformdata_left;

       }


modifica_dados_brutos(&amostras_
no_tempo_left[0],tamanho_da_janel
a);


modifica_dados_brutos(&amostras_
no_tempo_right[0],tamanho_da_jan
ela);

    for(long
i=0;i<tamanho_da_janela;i++)

    {

      waveformdata_left=(unsigned
char)amostras_no_tempo_left[i];


fwrite(&waveformdata_left,sizeof(wa
veformdata_left),1,fw);

      waveformdata_right=(unsigned
char)amostras_no_tempo_right[i];


fwrite(&waveformdata_right,sizeof(w
aveformdata_right),1,fw);

       }

     }

else
if((resolucao==16)&&(wave_chunk.n
umberofchannels==1))

  {

cout<<"\n\n16 bits 1 canal";

getch();

  unsigned char
waveformdata_lsb,waveformdata_m
sb;

  double* amostras_no_tempo =
new double[tamanho_da_janela];

  for(long
i=0;i<tamanho_da_janela;i++)

    {


fread(&waveformdata_lsb,sizeof(wa
veformdata_lsb),1,fr);


fread(&waveformdata_msb,sizeof(w
aveformdata_msb),1,fr);


amostras_no_tempo[i]=(double)con
verte2de8para1de16(waveformdata
_lsb,waveformdata_msb);

    }


modifica_dados_brutos(&amostras_
no_tempo[0],tamanho_da_janela);

  for(long
i=0;i<tamanho_da_janela;i++)
```

```
   {

converte1de16para2de8((short)(amostras_no_tempo[i]),&waveformdata_lsb,&waveformdata_msb);

fwrite(&waveformdata_lsb,sizeof(waveformdata_lsb),1,fw);

fwrite(&waveformdata_msb,sizeof(waveformdata_msb),1,fw);

   }}

else if ((resolucao==16)&&(wave_chunk.numberofchannels==2))

    {

//cout<<"\n\n16 bits 2 canais";

//getch();

 unsigned char waveformdata_lsb_left,waveformdata_lsb_right,waveformdata_msb_left,waveformdata_msb_right;

     double* amostras_no_tempo_left = new double[tamanho_da_janela];

     double* amostras_no_tempo_right = new double[tamanho_da_janela];

     cout<<"\n\nConvertendo 2 de 8 para 1 de 16";

     for(long i=0;i<tamanho_da_janela;i++)

        {

fread(&waveformdata_lsb_left,sizeof(waveformdata_lsb_left),1,fr);

fread(&waveformdata_msb_left,sizeof(waveformdata_msb_left),1,fr);

fread(&waveformdata_lsb_right,sizeof(waveformdata_lsb_right),1,fr);

fread(&waveformdata_msb_right,sizeof(waveformdata_msb_right),1,fr);
amostras_no_tempo_left[i]=(double)converte2de8para1de16(waveformdata_lsb_left,waveformdata_msb_left);
amostras_no_tempo_right[i]=(double)converte2de8para1de16(waveformdata_lsb_right,waveformdata_msb_right);

        }

modifica_dados_brutos(&amostras_no_tempo_left[0],tamanho_da_janela);

modifica_dados_brutos(&amostras_no_tempo_right[0],tamanho_da_janela);

        // cout<<"\n\nConvertendo 1 de 16 para 2 de 8";

         for(long i=0;i<tamanho_da_janela;i++)

           {
converte1de16para2de8((short)amostras_no_tempo_left[i],&waveformdata_lsb_left,&waveformdata_msb_left);

converte1de16para2de8((short)amostras_no_tempo_right[i],&waveformdata_lsb_right,&waveformdata_msb_right);

fwrite(&waveformdata_lsb_left,sizeof(waveformdata_lsb_left),1,fw);

fwrite(&waveformdata_msb_left,sizeof(waveformdata_msb_left),1,fw);
```

```
fwrite(&waveformdata_lsb_right,size
of(waveformdata_lsb_right),1,fw);


fwrite(&waveformdata_msb_right,siz
eof(waveformdata_msb_right),1,fw);
          }        }
      else
        {
       // cout<<"Resolucao ou
numero de canal(is) invalido(s)";
//getch();
        exit(0);
        }
      unsigned int c;

while((c=getc(fr))!=EOF)//termina de
gravar os cabecalhos de fim de
arquivo wav
          putc(c,fw);
     }
    else
       //out<<"\n\nFORA DO
FORMATO PCM...";
//getch();
       fclose(fr);
       fclose(fw);
    }
else
  cout<<"\n\nArquivo nao existe ou
nao pode ser aberto";
  cout<<"\n\n\n";
getch();
}
//-----------------------------------

short
converte2de8para1de16(unsigned
char lsb, unsigned char msb)
{
 return(((msb&0x80)>>7)*(32768)+
       ((msb&0x40)>>6)*(16384)+
       ((msb&0x20)>>5)*(8192)+
       ((msb&0x10)>>4)*(4096)+
       ((msb&0x08)>>3)*(2048)+
       ((msb&0x04)>>2)*(1024)+
       ((msb&0x02)>>1)*(512)+
       ((msb&0x01))*(256)+
       ((lsb&0x80)>>7)*(128)+
       ((lsb&0x40)>>6)*(64)+
       ((lsb&0x20)>>5)*(32)+
       ((lsb&0x10)>>4)*(16)+
       ((lsb&0x08)>>3)*(8)+
       ((lsb&0x04)>>2)*(4)+
       ((lsb&0x02)>>1)*(2)+
       (lsb&0x01));
       }
       //-------------------------------------
-
void converte1de16para2de8(short
resultado, unsigned char*
lsb,unsigned char* msb)
{

*lsb=(((resultado&0x0080)>>7)*(128
)+
      ((resultado&0x0040)>>6)*(64)+
```

```
((resultado&0x0020)>>5)*(32)+
((resultado&0x0010)>>4)*(16)+
((resultado&0x0008)>>3)*(8)+
((resultado&0x0004)>>2)*(4)+
((resultado&0x0002)>>1)*(2)+
((resultado&0x0001)));

*msb=(((resultado&0x8000)>>15)*(128)+

((resultado&0x4000)>>14)*(64)+

((resultado&0x2000)>>13)*(32)+

((resultado&0x1000)>>12)*(16)+
((resultado&0x0800)>>11)*(8)+
((resultado&0x0400)>>10)*(4)+
((resultado&0x0200)>>9)*(2)+
((resultado&0x0100)>>8));
      }
//----------------------------------------
void modifica_dados_brutos(double* dados, long tamanho)
{
    int fl,n;
    float N1,N2,N3;
    int amostras;
     amostras=400;
    N1=(0.1)*amostras;
    printf("N1=%f",N1);
    N2=(0.65)*amostras;
    printf("N2=%f",N2);
    N3=N1+N2;
    printf("N3=%f",N3);
    for(n=0;n<tamanho+1;n++)
    {fl=n;

if(fl>amostras){fl=fl%amostras;}//se fl for maior que o numero de amostras, divide por amostras e pega o resto
    if(fl<N1)
    {dados[n]=1000*(1/2)*(1-cos(3.1415*fl/N1));
                }
    else {if(fl<N3){dados[n]=1000*cos(3.1415*(fl-N1)/(2*N2));}
          else{dados[n]=0;}
          }  }
                    printf("\n\n");
                    for(int j =8001;j<10000;j++)
                    printf(" %ld,",dados[j]);
                    getch();

////////////////////////////////////////////////////
////////////////////////////////////////////////////
  /*  int m=29;
   float conv[m];
// int ju=m+tamanho-1;
   double dados_conv[m];
conv[0]= -0.000000;
conv[1]= -0.000000;
```

```
conv[2]= 0.000000;
conv[3]= -0.000000;
conv[4]= -0.000002;
conv[5]= -0.000000;
conv[6]= 0.000011;
conv[7]= 0.000000;
conv[8]= -0.000037;
conv[9]= 0.000000;
conv[10]= 0.000109;
conv[11]= 0.000000;
conv[12]= -0.000272;
conv[13]= -0.000000;
conv[14]= 0.000584;
conv[15]= -0.000000;
conv[16]= -0.000435;
conv[17]= -0.000000;
conv[18]= -0.001136;
conv[19]= 0.000000;
conv[20]= 0.005559;
conv[21]= -0.000000;
conv[22]= -0.013837;
conv[23]= -0.000000;
conv[24]= 0.025517;
conv[25]= 0.000000;
conv[26]= -0.038188;
conv[27]= 0.000000;
conv[28]= 0.048120;
conv[29]= -0.000000;
printf("\n\n\n\n
tamanho=%l",tamanho)  ;
 /*   getch();
            for(int s=0;s<=m;s++)
                    dados_conv[s]=0;
           // for(int f =99; f<=100;f++)
            //printf("\n\ndados[%d]=
%lf",f,dados[f]);
             for(int c =0; c<=m;c++)
            {
                    for(int
s=0;s<=100;s++)
                        {

if(s>=c)
                                        {

dados_conv[s]=dados_conv[s]+dad
os[s-c]*conv[c];
getch();
                                        }
                        }

printf("\n\ndados_conv[%d]=%lf",c,d
ados_conv[c]);                }
       printf("ACABOU!!!!!!!!!!");     */
}
```

7. Referências Bibliográficas

[1] Quatieri, T.F. Discrete-Time Speech Signal Processing: Principles and Practice.

New Jersey: Pretice-Hall, 2001

[2] A.V. Oppenhein e R.W. Schafer, “Discrete Time Signal Processing”, 2. ed., Prentice

Hall, New York, 1999.

[3] S. Haykin e B.V. Veen, “Sinais e Sistemas”, Bookman, Porto Alegre, 2001.

[4] Lyons,R.D. Understanding Digital Signal Processing. 2 ed. New Jersey: Prentice

Hall, 2004.

[5] Jain, A.; Hong, L.; Pankanti, S.Biometric Identifications. Communications of the

ACM. N.43, v.2, p.90-98,2000.

[6] Juang,B.H.; Chou,W. Pattern Recognition in Speech and Language Processing.

Boca Raton: CRC Press, 2003.

[7] Casanova,J.P, " Manual de fonoaudiologia", Porto Alegre: Artes Médicas, 1997.

[8] Kaplan HM. Anatomy and physiology of speech. New York McGraw-Hill, 1971.

[9] Deng, L.; O´Shaughnessy, O. Speech processing: a dynamic and optimization-oriented approach. Marcel Dekker Inc: New York -USA 2003.

[10]Dajer, Eugenia M. Padrões Visuais de Sinais de Voz Através de Técnicas de Análise Não Linear. São Carlos: EESC – USP, 2006.Tese (Mestrado).

[11] Barbosa, P.A.; Máquinas falantes como instrumento lingüístico: por um humanismo éclaire. Línguas e Instrumentos Lingüísticos 2001.

[12] Simões, Flávio O. Implementação de um Sistema de Conversão Texto-Fala para o Português do Brasil. Campinas: UniCamp. 1999. Tese (Mestrado).

[13] Barros, Maria J. A. Estudo Comparativo e Técnicas de Geração de Sinal para a Síntese da Fala. Porto: Faculdade de Engenharia da Universidade do Porto. 2002. Tese (Mestrado).