CENTRO UNIVERSITÁRIO SALESIANO DE SÃO PAULO

UNISAL – *CAMPUS* DOM BOSCO

Lucas Lui Motta

ALGORITMOS PARA REDUÇÃO DE RUÍDO IMPLEMENTADOS COM DSP

Americana

2016

Lucas Lui Motta

ALGORITMOS PARA REDUÇÃO DE RUÍDO IMPLEMENTADOS COM DSP

Trabalho de Conclusão de Curso apresentado como exigência parcial para obtenção do grau de Bacharel em Engenharia Elétrica, modalidade Eletrônica, no Centro Universitário Salesiano de São Paulo, unidade de Americana, sob a orientação do Prof. Dr. Fernando Oscar Runstein

Americana

2016

Dedico este trabalho a minha família pela superação e compreensão de cada um;
ao meu orientador que apresentou muita sabedoria e paciência na orientação;
a todos os professores e colegas que contribuíram para a minha formação
a qual foi necessária para a realização deste trabalho.

AGRADECIMENTOS

Agradeço primeiramente a minha família, constituída pelo meu pai, mãe e irmão. Ao meu pai, responsável pelo apoio e incentivo em todos os dias durante a elaboração desse trabalho, e durante a minha formação, lembrando-me sempre a importância daquilo que eu estava fazendo, e se esforçando para manter a família em ordem.

A minha mãe, pela sua força de vontade em permanecer feliz diante das dificuldades que a cercam, às vezes até se sacrificando para não me interromper durante os estudos. Agradeço também sua calma nos momentos em que estive estressado ou cansado demais para conversar.

A meu irmão, pelo suporte nos momentos em que estive ausente, auxiliando física e psicologicamente meus pais. Além disso, pela sua ajuda nas tarefas domiciliares que executou perfeitamente, para que eu pudesse ter mais tempo para os estudos.

Tenho muito a agradecer ao meu orientador Prof. Dr. Fernando Oscar Runstein, que me auxiliou de todas as formas possíveis durante a realização desse trabalho, compartilhando sua sabedoria com a devida maestria e *expertise* na didática, facilitando o desenvolvimento do trabalho e fazendo com que fosse, mais do que tudo, prazeroso elabora-lo, deixando assim uma semente que me impulsa a querer mais conhecimento e crescimento na vida acadêmica.

A todos os professores que contribuíram para minha formação em engenharia, do básico ao avançado da matemática, fazendo com que fosse possível realizar um trabalho teórico e prático como este.

Aos colegas de sala e de trabalho que direta ou indiretamente colaboraram para a conclusão deste trabalho, fazendo-me companhia na biblioteca ou possibilitando a minha ausência da empresa alguns dias chaves.

A minha família atento-me todo dia a retribuir esse amor. Aos meus professores espero um dia retribuir, seja com algum serviço, ou até ajudando a formar à próxima geração. A meus colegas de sala e trabalho, um esforço próprio será dado para agradecê-los.

RESUMO

Este trabalho foca o estudo de algoritmos atuais de redução de ruído e a implementação de um deles em DSP (*Digital Signal Processor* – Processador de Sinais Digitais) de forma que possa ser empregado em sistemas de controle de ruído em tempo real. O controle de ruído pode ser utilizado na redução de ruído em sinais de áudio e fala, o que permite melhorar a qualidade e inteligibilidade destes sinais em sistemas de comunicação, assim como em diversos outros equipamentos como, por exemplo, fones de ouvido, telefones celulares e sistemas de reconhecimento automático de fala.

Palavras-Chave: Redução de ruído. *Speech Enhancement*. Controle de ruído. DSP.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ADC: *Analog-to-Digital Converter*, do inglês, Conversor Analógico para Digital.

DAC: *Digital-to-Analog Converter*, do inglês, Conversor Digital para Analógico.

DCI: *Data Converter Interface*, do inglês, Interface Conversor de Dados.

DD: *Decision-Directed,* do inglês, Dirigido pela Decisão.

DFT: *Discrete Fourier Transform*, do inglês, Transformada Discreta de Fourier.

DR: *Dynamic Range*, do inglês, Faixa Dinâmica.

DSC: *Digital Signal Controller*, do inglês, Controlador de Sinal Digital.

DSP: *Digital Signal Processor*, do inglês, Processador de Sinais Digitais.

FFT: *Fast Fourier Transform*, do inglês, Transformada Rápida de Fourier.

FIR: *Finite Impulse Response*, do inglês, Resposta Impulsiva Finita.

I^2C: *Inter-Integrated Circuit*, do inglês, Circuito Integrado para Integrado.

IDE: *Integrated Development Environment*, do inglês, Ambiente de Desenvolvimento

LIT: *Linear Time-Invariant*, do inglês, Linear e Invariante no Tempo.

IFFT: *Inverse Fast Fourier Transform,* do inglês, Inversa Transforma Rápida de Fourier.

LMS*: Least Mean Square*, do inglês, Mínimo Quadrado Médio.

MAC: *Multiplier–Accumulator*, do inglês, Multiplicador e Acumulador.

NLMS*: Normalized Least Mean Square*, do inglês, Mínimo Quadrado Médio Normalizado.

RAM: *Random Access Memory*, do inglês, Memória Acesso Aleatório.

RLS: *Recursive Least* Square, do inglês, Mínimo Quadrado Recursivo.

SNR: *Signal-to-Noise Ratio*, do inglês, Relação Sinal-Ruído.

SNR: *Signal-to-Noise Ratio, do inglês, Relação Sinal-Ruído.*

SPI: *Serial Peripheral Interface*, do inglês, Periférico Serial Interface.

SSNR: *Segmental SNR*, do inglês, Relação Sinal-Ruído Segmentada.

USB: *Universal Serial Bus*, do inglês, Barramento Universal Serial.

VAD: *Voice Activity Detection, do inglês*, Detecção de Atividade de Voz.

WGN: *White Gaussian Noise*, do inglês, Ruído Branco Gaussiano.

SUMÁRIO

1. INTRODUÇÃO

Os algoritmos para redução de ruído em sinais de áudio permitem melhorar a qualidade e inteligibilidade destes sinais. Com sinais de melhor qualidade é possível aumentar o índice de acertos de um sistema de reconhecimento de fala (sistema que reconhece o que uma pessoa disse analisando a fala emitida), diminuir a incerteza quanto à identidade de uma pessoa em sistemas de verificação de locutor pela voz (biometria por voz: identificar ou verificar a identidade de uma pessoa analisando as características de sua voz), e aumentar a satisfação do usuário em sistemas de comunicação (telefonia fixa ou celular, videoconferência, *chats* etc, onde uma voz inteligível e de boa qualidade ajuda à sua compreensão). Existem, ainda, sistemas para controle ativo de ruído em tempo real, que permitem atenuar fortemente o ruído ambiente somando um ruído igual e de sinal contrário ao existente. Esta tecnologia pode ser utilizada em fones de ouvido, no interior de veículos para minimizar o ruído externo, e até em ambientes industriais, onde há níveis de ruído elevado vindo de centrífugas, prensas, máquinas ferramentas, pontes rolantes, ventiladores etc.

1.1. OBJETIVOS GERAIS

O objetivo deste trabalho é estudar algoritmos de redução de ruído e implementar um deles em DSP de forma que possa ser empregado em sistemas de controle de ruído em tempo real, trazendo o benefício desta tecnologia à sociedade em geral e a quem trabalha em ambientes ruidosos em especial.

1.2. OBJETIVOS ESPECÍFICOS

- Estudar algoritmos de redução de ruído para sinais de áudio e fala.
- Analisar o comportamento dos algoritmos em *software* MATLAB.
- Analisar sistemas com DSP que permitam a implementação dos algoritmos de redução de ruído (características, linguagem de programação, facilidade de uso e custo) e definir o que será utilizado no projeto.
- Implementar o algoritmo de redução de ruído no DSP e testar seu comportamento com diversos sinais ruidosos para avaliar sua eficiência, tanto do ponto de vista da implementação como da redução de ruído.

1.3. JUSTIFICATIVA

Diversas tecnologias atuais trabalham com sinais de áudio e/ou fala, e é necessário minimizar o ruído que degrada estes sinais de forma a melhorar a sua qualidade e inteligibilidade. O processamento desses sinais deve ser feito muitas vezes em tempo real pelo que é necessário o uso de DSPs, que permitem altíssima velocidade de processamento.

Algoritmos eficientes para redução de ruído implementados em DSPs podem contribuir para melhorar as taxas de acerto de sistemas de reconhecimento de fala e de reconhecimento de locutor, e aumentar a qualidade dos sistemas de comunicações utilizados pela sociedade.

Com o controle ativo de ruído também é possível construir dispositivos como fones de ouvido capazes de atenuar os ruídos do ambiente e reproduzir apenas o sinal de interesse, tornando o ambiente mais propício para comunicação entre os usuários destes dispositivos. Esse tipo de aplicação do algoritmo de redução de ruído pode beneficiar colaboradores de usinas ou siderúrgicas que necessitam se comunicar dentro de ambientes ruidosos.

1.4. SÍNTESE BIBLIOGRÁFICA

Artigo 1: Speech Enhancement using Spectral Subtraction-type Algorithms: A Comparison and Simulation Study – KARMAKAR, Abhijit; UPADHYAYA, Navneet. Publicado em: Procedia Computer Science, 2015.

Um dos primeiros algoritmos para redução de ruído propostos é o de subtração espectral o qual é proposto para sinais de vozes de único canal. Esse algoritmo consiste em estimar padrões espectrais do ruído em períodos onde não está presente a fala ou o áudio de interesse, e essa estimação é subtraída do sinal ruidoso para eliminar ou reduzir o ruído. O problema geral desses algoritmos de subtração espectral é a introdução de distorções que pode perceber-se como um ruído de fundo conhecido como ruído musical (do inglês, *musical noise*).

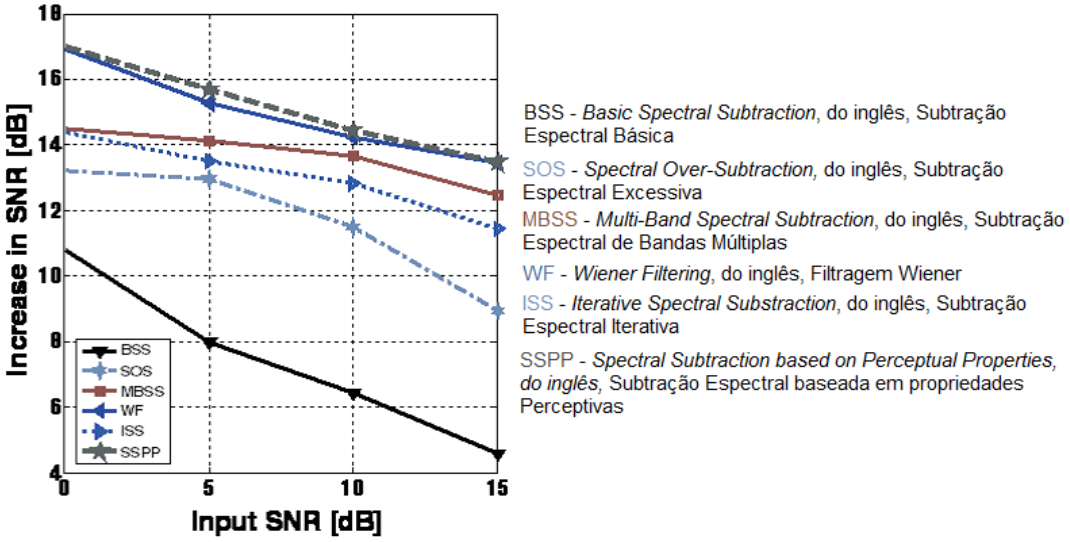
Em função das diferentes necessidades e aperfeiçoamentos, foram desenvolvidas várias versões de algoritmos de subtração espectral, entre essas o autor destaca o de subtração espectral por excesso de subtração, subtração espectral de bandas múltiplas, subtração espectral pelo mínimo erro quadrático (subtipo baseado no filtro de Wiener), subtração espectral iterativa e subtração espectral baseada em propriedades perceptivas.

Neste artigo cada algoritmo passou por testes objetivos (SNR e PESQ – *Perceptual Evaluation of Speech Quality*) e testes subjetivos (opinião de ouvintes baseada na compreensão ou inteligibilidade da fala). Os testes foram executados com sinais ruidosos estacionários e não estacionários.

A figura 1 mostra os resultados dos testes com os diferentes algoritmos de subtração espectral em função da SNR do sinal de entrada e a SNR do sinal filtrado.

O artigo conclui que o algoritmo clássico de subtração espectral teve a menor qualidade de filtragem e introduz um ruído de fundo que pode diminuir a inteligibilidade da voz, por outro lado, o que mais se mostrou eficiente foi o baseado em propriedades perceptivas.

Figura 1 – O valor de SNR para os diferentes tipos de subtração espectral para WGN



Fonte: Adaptado do artigo (UPADHYAY; KARMAKAR, 2015).

Artigo 2: Comparison between Adaptive filter Algorithms (LMS, NLMS and RLS) – DHIMAN, Jyoti; GHMAD, Shadab; GULIA, Kuldeep. Publicado em: International Journal of Science - 2013.

O artigo traz uma comparação entre a implementação de três algoritmos de filtragem adaptativa para redução de ruído, são eles: *Least Mean Square* (LMS), *Normalized Least Mean Square* (NLMS) e *Recursive Least* Square (RLS).

Esses algoritmos convergem para o estado de “filtro ótimo” a todo instante encontrando os coeficientes do filtro que relaciona o erro quadrático médio mínimo da diferença do sinal desejado com o sinal atual. Como os próprios autores citam, esses algoritmos utilizam poucas amostras de entrada e saída e por isso são geralmente utilizados em sistemas de tempo real.

Nos testes de desempenho, os algoritmos foram testados nos quesitos de rapidez de convergência para o erro quadrático médio mínimo (MMSE - *Minimum Mean Square Error*), complexidade computacional (ordem do filtro gerada) e na estabilidade da aplicação diante de diversos tipos de sinais. A tabela 1 a seguir mostra os resultados desse teste.

Tabela 1 - Valores encontrados nos testes de desempenho dos algoritmos.

S. No.	Algorithms	MSE	Complexity	Stability
1.	LMS	$1.5*10^{-2}$	2N+1	Less Stable
2.	NLMS	$9.0*10^{-3}$	3N+1	Stable
3.	RLS	$6.2*10^{-3}$	$4N^2$	High Stable

Fonte: DHIMAN; GHMAD (2013).

O artigo também propôs a comparação dos resultados da filtragem utilizando os algoritmos; os resultados são mostrados na tabela 2. O autor conclui que o melhor algoritmo adaptativo foi o RLS nos diversos quesitos (SNR, desempenho, rapidez de convergência e estabilidade).

Tabela 2 - Valores de SNR encontrados na comparação dos algoritmos de filtros adaptativos.

Noise Variance	Sampling Rate (kHz)	SNR Improvement (dB) LMS	SNR Improvement (dB) NLMS	SNR Improvement (dB) RLS
0.02	1.5	8.85	9.85	9.91
0.05	1.5	7.55	8.62	8.89
0.10	1.5	5.12	6.38	7.02

Fonte: DHIMAN; GHMAD (2013).

Artigo 3: Subjective Comparison of Speech Enhancement Algorithms – HU, Yi; LOIZOU, Philipos C. Publicado em: International Conference on Acoustics Speech and Signal Processing Proceedings, 2006.

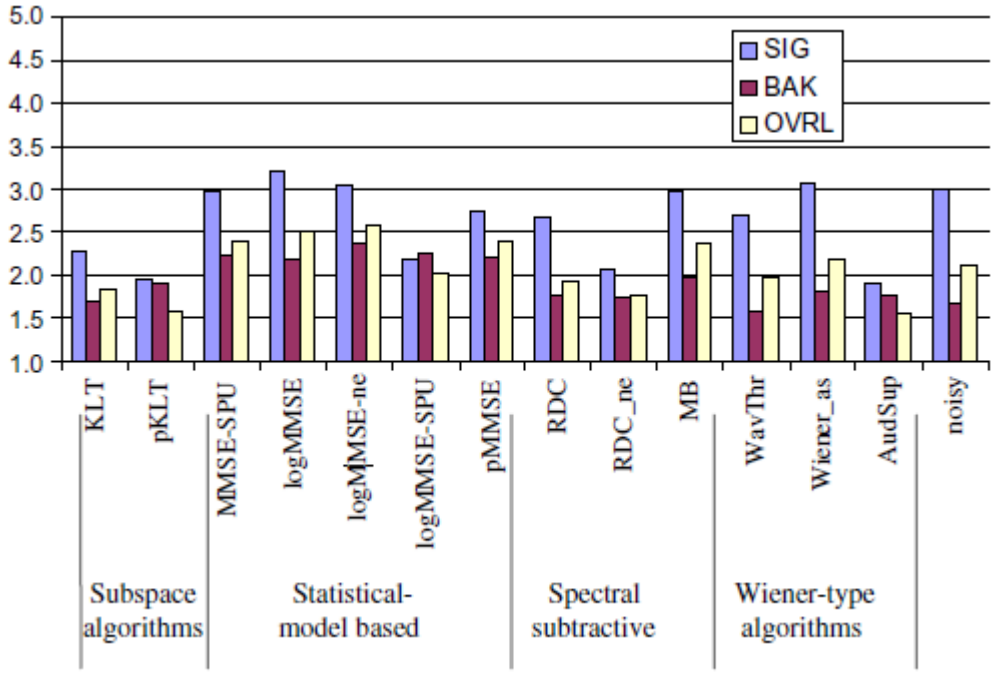
Neste artigo, bastante citado na literatura, houve uma comparação baseada em testes subjetivos do desempenho de vários algoritmos desenvolvidos no decorrer do tempo para melhoramento de sinais de voz.

Os autores utilizaram para a comparação vários sinais de voz com diferentes tipos de ruídos e no total foram testados treze algoritmos divididos em quatro classes: *Spectral Subtractive*, *Subspace*, *Statistical-Model Based* e *Wiener-type*.

A figura 2 mostra o resultado dos testes feitos em cima de um sinal de voz contaminado com ruído ambiente vindo da rua (*street noise*).

.

Figura 2 - Resultados dos 13 algoritmos testados na filtragem de um ruído.



Fonte: HU; LOIZOU (2016).

O teste foi realizado pelos autores com a ajuda de 32 ouvintes, onde SIG, BAK e OVRL, são respectivamente: distorção da voz, percepção do ruído e qualidade geral do áudio. Cada recebe uma nota de 1 a 5 de possíveis, sendo 5 a melhor nota e 1 a pior.

Entre as conclusões dos autores tem-se que, em termos de qualidade geral e distorção de voz, os melhores algoritmos foram os baseados em modelamento estatístico, em seguida ficaram os baseados em subtração espectral e filtros de Wiener, e por último os algoritmos baseados em *subspace*.

2. DESENVOLVIMENTO

Para implementar em DSP um sistema de redução de ruído em sinais de voz (ou áudio em geral), é necessário, primeiramente, converter o sinal de interesse em um sinal digital e logo processar esse sinal no DSP em que está implementado o algoritmo de redução de ruído. A figura 3 mostra isto.

Figura 3 - Desenho ilustrativo da funcionalidade da implementação.

Fonte: própria.

A seguir serão explicadas as características dos sinais de voz e áudio, tipos de ruídos esperados, a conversão analógico/digital e, por último, a forma em que será feita a análise e tratamento do sinal.

2.1. SINAIS DE ÁUDIO E VOZ

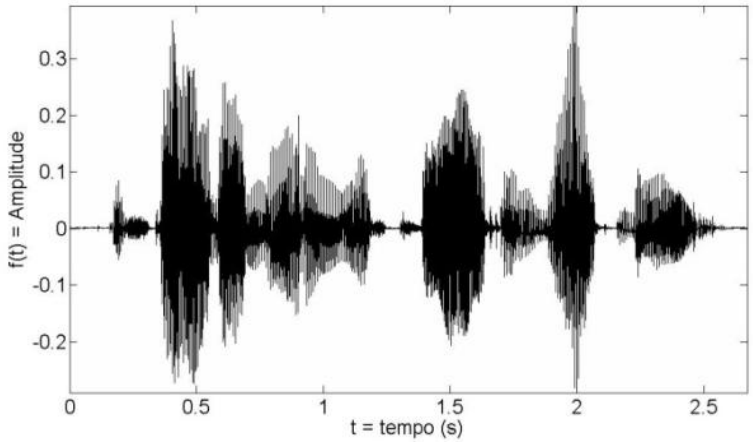
Sinais são variáveis que carregam informação, em geral, sobre o estado ou o comportamento de um fenômeno físico. Sinais são representados matematicamente por funções de uma ou mais variáveis (OPPENHEIM; SCHAFER, 2009).

Ondas sonoras são um exemplo de fenômeno físico que pode ser representado por um sinal, essas ondas são criadas quando um material vibra, e essa vibração força as moléculas de ar a sua volta a se comprimirem, criando regiões com maior e menor pressão. Sinais de áudio são formados por ondas sonoras na faixa de frequência audível, por exemplo, sons que se ouve de máquinas, instrumentos musicais e ambientes; definem-se sinais de áudio aqueles cujas componentes de frequências vão de 20 a 20kHz (LI; DREW, 2004).

Existem sinais unidimensionais, bidimensionais e tridimensionais, isto é, são funções de uma, duas ou três variáveis independentes respectivamente. O sinal de áudio é um exemplo de sinal unidimensional, pois sua amplitude pode representar-

se em função do tempo (uma variável independente), conforme é mostrado na figura 4 (OLIVEIRA; ANDRADE, 2006).

Figura 4 - Gráfico da função de um sinal de áudio no tempo.



Fonte: própria.

Um sinal de áudio contendo fala é chamado de sinal de fala (ou de voz) e suas componentes de frequência vão, tipicamente, até 4[kHz]. A produção da fala envolve um conjunto de órgãos e músculos que inclui pulmões, laringe e trato vocal (boca, língua, lábios, glote etc) (LI; DREW, 2004).

Outra característica importante de sinais de fala é o fato de não serem estacionários, ou seja, suas componentes espectrais variam com o tempo, mas o fazem lentamente. Desta forma, em períodos de tempo pequenos (10 a 30 [ms]) a fala pode ser considerada estacionária. Isso torna possível efetuar sua análise e filtragem nesses intervalos de tempo como será discutido mais adiante (LOIZOU, 2013).

2.2. TIPOS DE RUÍDOS

Dependendo do cenário, os sinais de voz podem sofrer degradações devido a ruídos de diversos tipos que provocam a diminuição de sua inteligibilidade e qualidade. Há diversos sistemas e algoritmos capazes de combater esses ruídos, mas é importante entender a fonte e características dos mesmos. A seguir são descritos alguns tipos de ruídos encontrados em sinais de áudio. Mais detalhes podem ser encontrados em (VASEGHI, 2008).

- Ruído acústico aditivo: compreende os diversos sons do ambiente, podendo ser originado por um fenômeno natural (ex. vento), uma máquina (ex. lava-roupas) ou até por outras vozes (ex. restaurante).

- Ruídos convolucionais: gerados pelo sistema de captura de voz (microfones) e/ou por um canal de comunicação. Seu efeito é representado no mundo digital pela convolução dos mesmos com a resposta impulsiva do sistema.

- Ruídos acústicos por reverberação: gerados pela reflexão do som original nas paredes, tetos e objetos ambientais; as reflexões se somam com atraso ao sinal original degradando-o. O efeito é comumente chamado de eco e ocorre em grandes galerias e salões.

- Ruídos causados pelos algoritmos de codificação com perdas de entropia do sinal original.

- Ruídos causados pelo equipamento eletrônico (pré-amplificador, amplificador equalizador etc), devidos à saturação ou ao controle automático de ganho durante a captura do sinal.

- Ruídos originados por perdas na transmissão do sinal de áudio, como por exemplo, perdas de pacotes em transmissões via rede *internet*.

Em termos das características temporais e espectrais, os ruídos podem ser estacionários, ou seja, seu espectro não muda ao longo do tempo (ex. ruído de um ventilador), ou não estacionários, no qual ocorrem mudanças de espectro ao longo do tempo (ex. um bar cheio de pessoas conversando). Podem ter uma largura de banda curta, ou seja, concentram a maior parte da energia do ruído em frequências específicas (ex. máquinas giratórias), ou conter uma banda larga onde tem energia na maioria das componentes do espectro (ex. ruído vindo de uma explosão).

A relação sinal-ruído (SNR - *Signal-to-Noise Ratio*) é uma medida objetiva que nos diz o quanto de ruído está presente em um sinal. Quanto maior o valor da SNR, menor é a presença do ruído no sinal. Essa relação é definida pelo logaritmo

decimal da razão da potência do sinal de voz e a potência do ruído, conforme equação (1.a). A equação (1.b) permite calcular a SNR para sinais que foram melhorados com algoritmos de redução de ruído.

$$SNR_{db} = 10\log_{10}\left(\frac{P_{sinal}}{P_{ruído}}\right) \quad (1.a)$$

$$SNR_{db} = 10\log_{10}\left(\frac{(x[n])^2}{(x[n]-y[n])^2}\right) \quad (1.b)$$

Onde: $x[n]$ - É o sinal original limpo.

$y[n]$ – É o sinal melhorado (filtrado).

As potências do sinal e do ruído devem ser medidas na mesma largura de análise temporal e espectral, ou seja, devem ser analisadas na mesma duração de tempo e banda de frequências. Uma medida mais precisa da relação entre a potência do sinal e a do ruído, é a relação sinal-ruído segmentada (SSNR – *Segmental* SNR), pois ela corresponde à média aritmética da SNR calculada em segmentos curtos do sinal de voz, de forma a equilibrar as diferenças entre os trechos de maior e menor intensidade do sinal. O cálculo da SSNR é dado pela equação [2] (LOIZOU, 2013).

$$SSNR_{db} = \frac{10}{M}\sum_{m=0}^{M-1}\log_{10}\left(\frac{\sum_{n=Nm}^{Nm+n-1}(x[n])^2}{\sum_{n=Nm}^{Nm+n-1}(x[n]-y[n])^2}\right) \quad (2)$$

Onde: N – É o comprimento das janelas de análise (valores entre 15–20 [ms]).

M – É o número de janelas analisadas do sinal.

2.3. CONVERSÃO ANALÓGICO/DIGITAL

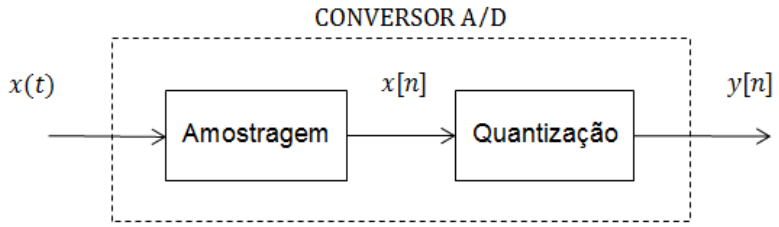
Sinais de áudio podem ser armazenados, transmitidos e processados. Dependendo do sistema adotado para estas operações, é necessário converter o sinal de analógico para digital ou vice-versa (OLIVEIRA; ANDRADE, 2006).

- Sinais analógicos: também chamados de contínuos em tempo e amplitude, são aqueles onde a função matemática que os descreve é definida para todos os valores de tempo (variável independente) e a amplitude (função do tempo) pode tomar qualquer valor dentro de determinado intervalo.

- Sinais digitais: são sinais discretos no tempo e amplitude, ou seja, a função matemática que os descreve é definida apenas para alguns valores da sua variável independente, enquanto que a amplitude pode assumir, apenas, um conjunto finito de valores.

O dispositivo responsável pela conversão do sinal de analógico para digital é o conversor A/D (ADC - *Analog-to-Digital Converter*), e o responsável pela conversão de digital para analógico é o conversor D/A (DAC - *Digital-to-Analog Converter*). O conversor A/D é composto por dois blocos conforme mostra a figura 5.

Figura 5 – Diagrama em Blocos do Conversor A/D.



Fonte: própria.

O bloco de amostragem recebe o sinal analógico *x*(t) e obtêm amostras dele em intervalos de tempo igualmente espaçados, tornando-o um sinal discreto no tempo *x*[n]. As amostras geradas são enumeradas por números inteiros, conforme equações [3.a] e [3.b].

$$x[n] = x_{am}(t) \qquad 0 < t < D_{segundos} \tag{3.a}$$

$$x[n] = x_{am}(nT_S) \qquad 0 < n < \left[\left(\frac{D_{segundos}}{T_S}\right) - 1\right] \tag{3.b}$$

Onde: n - Número inteiro que representa o índice da amostra.
Ts - Tempo de amostragem, que é a recíproca da freq. de amostragem.

Para que o processo de amostragem ocorra sem perda de informação e o sinal possa ser recuperado a partir de suas amostras, a frequência de amostragem deve ser no mínimo duas vezes maior que a máxima frequência do sinal original, conforme teorema de Nyquist-Shannon (NYQUIST, 1928; SHANNON, 1949).

O bloco de quantização torna o sinal *x*[n] recebido pelo bloco de amostragem em um sinal *y*[n] discreto em amplitude. Para isso, os valores de amplitude são mapeados em um conjunto finito de valores chamados níveis de quantização. Cada amplitude de entrada é alocado ao nível de quantização mais próximo. O número de níveis é dado por 2^B, onde B é o número de bits do quantizador (um quantizador de 8 bits terá, por tanto, 256 níveis de quantização disponíveis). Quanto maior B, maior a resolução do quantizador e menor o erro de quantização, definido como metade da distância entre 2 níveis de quantização.

Define-se a resolução de um conversor como:

$$N° \; de \; N.Q = 2^{N° \; de \; bits \; do \; Quantizador} \tag{4.a}$$

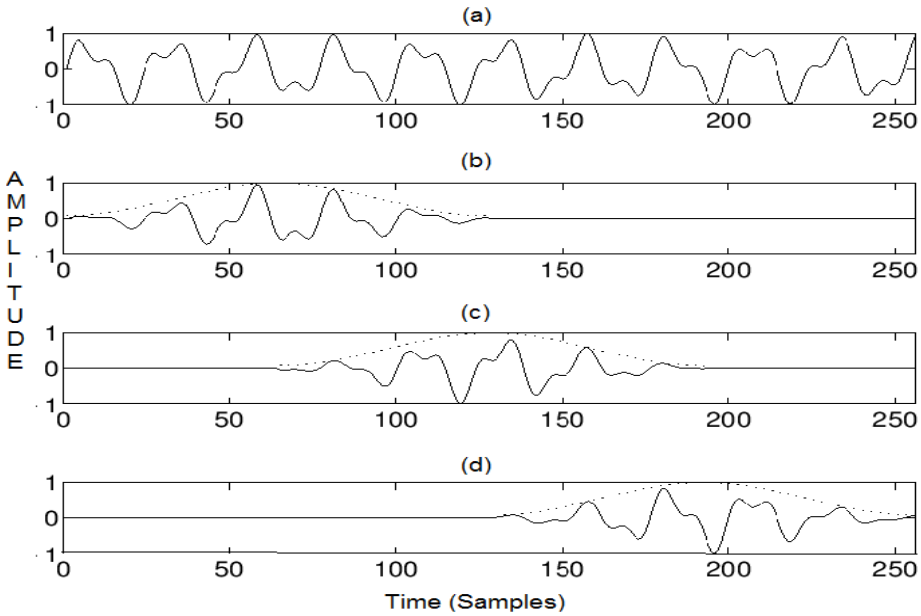
$$Resolução \; do \; Quantizador = \frac{Faixa \; de \; Amplitude}{N° \; de \; N.Q - 1} \tag{4.b}$$

Sendo assim, quanto maior o número de níveis de quantização, menor será a resolução do conversor e melhor é a representação das amplitudes do sinal original analógico no mundo digital.

2.4. ANÁLISE DE SINAIS DE ÁUDIO

Quando se analisam sinais de áudio em tempo real, escolhem-se quadros de análise de 10 a 30 [ms] de duração, de forma que o sinal pode ser considerado estacionário nesse intervalo. Esta análise é repetida a cada N [ms], escolhendo-se N, em geral, da metade da duração de um quadro de forma que existirá a sobreposição do final de um quadro com o início do próximo. Os quadros são ponderados para mudar sua forma de retangular para um formato próximo da função amostragem ou Sa(t). A figura 6 mostra um exemplo de janelamento de sinal de áudio utilizando a janela de Hamm.

Figura 6 - Janelamento com sobreposição de quadros em 50% (e janela do tipo Hamm).



Fonte: Adaptado de Smith (2011).

Cada quadro de amostras é obtido multiplicando o sinal de áudio *x*[n] pela janela de análise *w*[n] *deslocada*, conforme equação [5] abaixo (SMITH, 2011). A duração da janela *w*[n] limita a duração do sinal *x*[n] e modifica as amplitudes das amostras (essa modificação será explicada mais a frente).

$$x_m[n] = x[n].w[n - mR] \qquad 0 \leq n \leq Q - 1 \qquad (5)$$

Onde: m – Índice que indica o número do quadro.

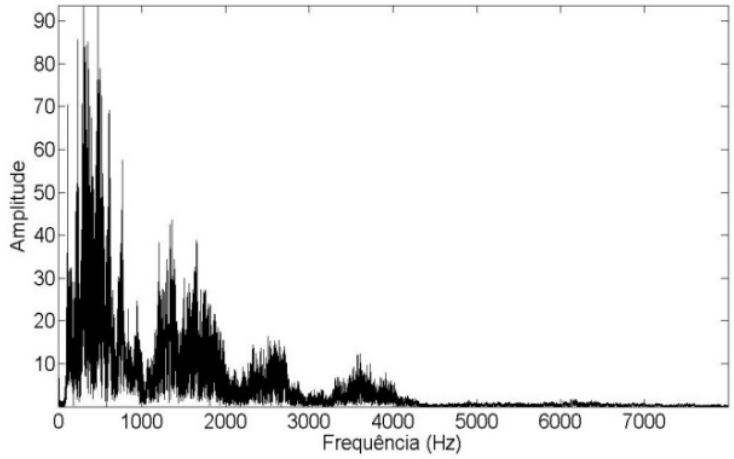
Q – Tamanho do quadro ou janela (em amostras).

R – Quantidade de amostras para o próximo quadro de analise (Passo de Analise).

A informação geralmente utilizada na análise de sinais de áudio é as componentes de frequência do mesmo. Para determinar essas componentes utiliza-se a Transformada de Fourier. No mundo digital a Transformada de Fourier é um sinal contínuo o que impede de ser tratado em um DSP, por isso, utilizam-se amostras desta transformada, que sim constituem um sinal discreto. O cálculo das amostras da Transformada de Fourier recebe o nome de Transformada Discreta de Fourier (DFT – *Discrete Fourier Transform*). Existem algoritmos rápidos para o cálculo da DFT chamados de FFT (*Fast Fourier Transform*). A representação de um

sinal no domínio da frequência é chamada de espectro de frequências do sinal e nele é possível ver a amplitude de cada componente de frequência, conforme figura 7.

Figura 7 – Espectro de frequências de um sinal de voz calculado pela FFT.



Fonte: própria.

Quando é feito o janelamento de um sinal, a janela de análise *w*[n] de regra não tem um formato retangular, já que isso implicaria no mundo da frequência, em multiplicar o espectro por uma função *Sampling*, o que deformaria o espectro original do sinal. Para não modificar o espectro do sinal, a janela de análise deve ser retangular no mundo da frequência, e isto implica em janelas com formato de *Sampling* no mundo do tempo. Sabendo disso, diversas janelas são encontradas na literatura com estas características; sendo as mais conhecidas as de Hamm (também conhecida como Hamming – que significa "janelamento com a janela de Hamm"), Hann, Blackman e Barlett. A função que define a janela de Hamm pode ser vista a seguir (equação [6]) (LOIZOU, 2013).

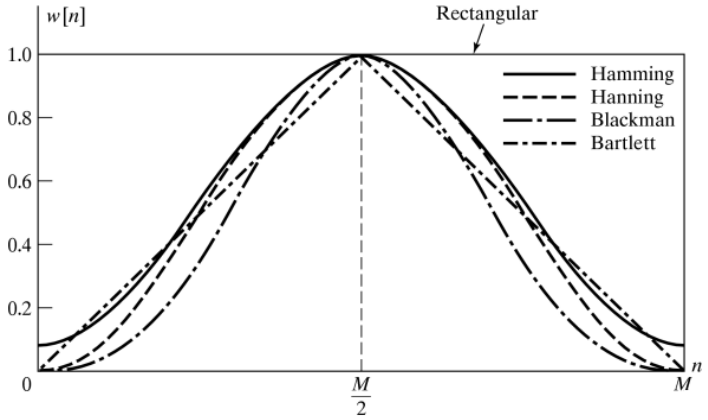
$$w[n] = \begin{cases} 0{,}54 - 0{,}46 \,.\cos\left(\dfrac{2\pi.n}{Q-1}\right) & 0 \le n \le Q-1 \\ 0 & Outros\ valores\ de\ 'n' \end{cases} \qquad (6)$$

Onde: n – É o índice das amostras do quadro.

Q – Tamanho do quadro ou janela (em amostras).

As funções das janelas antes mencionadas se diferenciam pela largura do lóbulo principal e o nível de lóbulo lateral (conforme figura 8), onde cada uma tem uma resultante espectral diferente (OPPENHEIM; SCHAFER, 2009).

Figura 8 – Janelas de análise frequentemente utilizadas.



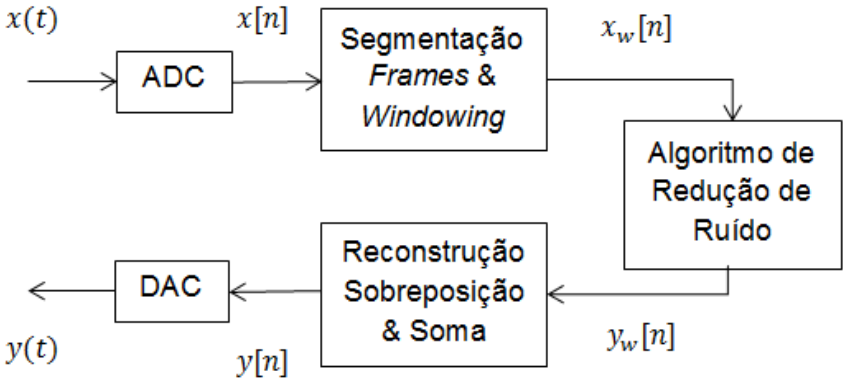
Fonte: Oppenheim; Schafer (2009, p. 536).

Posteriormente, para reconstruir o sinal de áudio processado utiliza-se a técnica *overlap and add* (sobreposição e soma) dos quadros obtidos. A equação a seguir demonstra como é feita a reconstrução em casos de sobreposição de 50% entre as janelas de análise.

$$x[n] = \sum_{m=0}^{L-1} x_m[n] = \sum_{m=0}^{L-1} x[n]\, w[n - mR] \tag{7}$$

Em termos práticos, pode ser visto na figura 9 o diagrama geral de análise e processamento de um sinal de áudio que será usado neste trabalho para aplicar os algoritmos de redução de ruído.

Figura 9 – Diagrama do processo de segmentação e reconstrução do sinal de áudio.



Fonte: própria.

3. ALGORITMOS DE REDUÇÃO DE RUÍDO

Como mencionado, diversos algoritmos têm sido propostos ao longo do tempo a fim de reduzir ou eliminar alguns dos ruídos mencionados. Neste projeto o foco será o tratamento de sinais degradados por ruídos acústicos aditivos, descorrelacionados com o sinal de voz (pressuposto razoável quando voz e ruído são de fontes independentes; VASEGHI, 2008). Para os outros tipos de ruídos como o ruído convolucional e o de reverberação, há outras técnicas na literatura melhor recomendadas como a Normalização da Média Cepstral (CMN – *Cepstral Mean Normalisation*).

A seguir serão descritos três algoritmos clássicos para redução de ruído em sinais de voz: Subtração Espectral (SS – *Spectral Subtractive*), Filtro de Wiener (WF – *Wiener Filter*) no domínio do tempo e na frequência, e finalmente o algoritmo baseado em modelos estáticos o Erro Quadrático Médio Mínimo (MMSE - *Mininum Mean Square Error*).

Os três algoritmos assumem que o sinal de voz ruidoso é a soma do sinal de voz limpo e o ruído (equação [8]) e são fundamentados no pressuposto de que para intervalos de tempo curtos, o sinal de áudio é considerado estacionário (LOIZOU, 2013).

$$x[n] = y[n] + n[n] \tag{8}$$

Onde: x[n] – Sinal de voz ruidoso.

y[n] – Sinal de voz.

n[n] – Ruído.

Além disso, os algoritmos propostos podem ser utilizados em aplicações contendo um único microfone de captura do áudio. Essa configuração da origem a uma preocupação, a de estimar o ruído devido à ausência de um microfone próprio para sua captura. Sendo assim, em todos os algoritmos citados, o ruído é estimado em tempos de silêncio, ou seja, tempos em que não há fala, seja utilizando um Detector de Atividade de Voz (VAD - *Voice Activity Detection)* ou assumindo um tempo inicial do áudio como sendo de silêncio e assim estimando o ruído no mesmo.

3.1. SUBTRAÇÃO ESPECTRAL

O algoritmo de subtração espectral explora o fato que o ruído é aditivo e, por tanto, pode obter-se o sinal de voz limpo subtraindo o espectro do ruído do espectro do sinal ruidoso. Isto implica em trabalhar no domínio da frequência. Sendo assim, aplicando a transformada discreta de Fourier para ambos os lados da equação [8], é originada a expressão a seguir.

$$X[\omega] = Y[\omega] + N[\omega] \tag{9}$$

Onde: $X[\omega]$ – Espectro do sinal de voz ruidoso.

$Y[\omega]$ – Espectro do sinal de voz limpo.

$N[\omega]$ – Espectro do ruído aditivo.

Expressando a equação em função da subtração espectral dos termos $X[\omega]$ e $N[\omega]$ e adotando a forma polar dos membros, temos a equação [10], onde pode adotar-se, em função de algumas simplificações (LOIZOU, 2013) que a fase resultante da subtração dos termos pode ser a do sinal de entrada, conforme equação [11]:

$$Y[\omega] = [|X[\omega]| - |N[\omega]|]\, e^{j[\Phi_X(\omega) - \Phi_N(\omega)]} \tag{10}$$

$$Y[\omega] = [|X[\omega]| - |N[\omega]|]\, e^{j\Phi_x(\omega)} \tag{11}$$

O método de subtração espectral também pode utilizar o espectro de potências ao invés do espectro de amplitude dos sinais, o que dá origem a equação geral [12].

$$\left|\hat{Y}[\omega]\right|^{\beta} = |X[\omega]|^{\beta} - \left|\hat{N}[\omega]\right|^{\beta} \tag{12}$$

Onde: $\beta = 1$ – Subtração espectral dos espectros de amplitude dos sinais.

$\beta = 2$ – Subtração espectral dos espectros de potência dos sinais.

Conforme dito anteriormente, os algoritmos propostos dependem da estimava do ruído, a que é feita nos intervalos de silêncio. Estes intervalos podem ser

identificados utilizando um VAD (*Voice Activity Detector*) de forma a atualizar a estimativa periodicamente. Neste trabalho será assumido que os primeiros 100[ms] do sinal de áudio são de silêncio e será feita a estimativa de ruído calculando a média dos espectros nas janelas de análise desse intervalo conforme equações a seguir.

$$\left|\widehat{N}[\omega]\right|_{T}^{\beta} = \alpha \left|\widehat{N}[\omega]\right|_{T-1}^{\beta} + (1-\alpha)\,|X[\omega]|_{T}^{\beta} \tag{13}$$

$$\left|\widehat{N}[\omega]\right|^{\beta} = \frac{1}{M}\sum_{J=0}^{M-1}\left|X_{TS(J)}[\omega]\right|^{\beta} \tag{14}$$

Onde: α – É o parâmetro que controla a taxa de atualização do ruído.

TS – Espectro de potência/amplitude obtido no tempo de silêncio.

M - Número de janelas do tempo de silêncio.

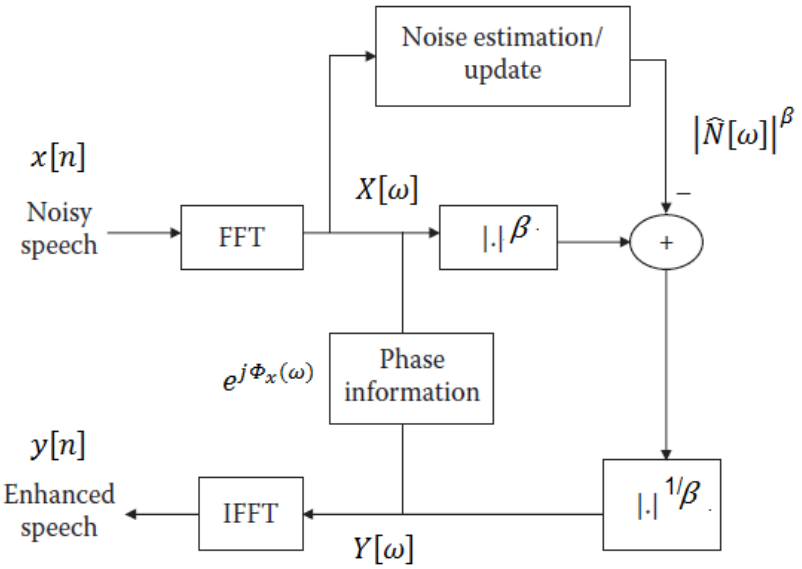
J - Número atual da janela.

Como a estimativa do ruído é baseada em uma média e constantemente atualizada, para não ficar à mercê de valores negativos do espectro do sinal filtrado, é definida uma expressão chamada de retificação de meia-onda (no método clássico de subtração espectral), conforme equação abaixo.

$$|X[\omega]| = \begin{cases} |Y[\omega]| - |N[\omega]|, & se \;\; |Y[\omega]| > \; |N[\omega]| \\ 0 & se\ não \end{cases} \tag{15}$$

Assim, quando a estimativa do ruído tiver valores maiores que o próprio espectro do sinal de voz ruidoso, a estimativa é zerada. O diagrama de blocos da figura 10 mostra a visão macro de trabalho deste algoritmo.

Figura 10 – Diagrama em Blocos do Algoritmo de Subtração Espectral.

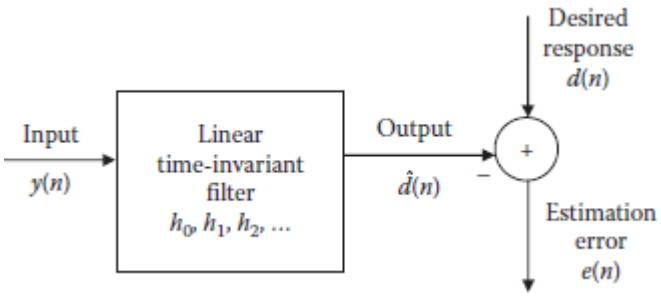


Fonte: Adaptado de Loizou (2013, p.96).

3.2. FILTRO DE WIENER

O algoritmo do filtro de Wiener é utilizado para redução de ruído em diferentes tipos de sinais como de imagem e áudio. O princípio básico do algoritmo é obter uma estimativa do sinal de voz limpo através do sinal de voz ruidoso. Essa estimativa é obtida com a ajuda do método de minimização da média quadrática do erro entre o sinal desejado e o sinal estimado (MSE- *Mean Squarer Error*). Seu funcionamento genérico pode ser visto na figura 11.

Figura 11 - Diagrama genérico do filtro de Wiener



Fonte: (LOIZOU, 2013).

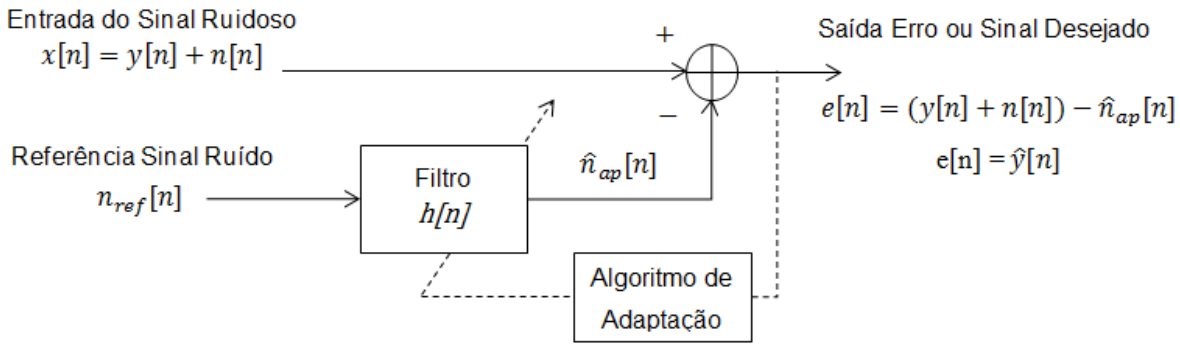
Dado um determinado filtro linear e invariante no tempo (LTI –*Linear Time-Invariant*) é possível estimar a resposta impulsiva necessária para tornar o sinal de entrada *y*[n] semelhante ao sinal desejado *d*[n], fazendo tão pequena quanto

possível a diferença do sinal de saída do sistema com o sinal desejado. A essa solução é dado o nome de filtro de Wiener, pois o primeiro a formular e resolver o filtro ótimo que minimiza esse erro estimado foi Norbert Wiener, e o mesmo pode ser executado no domínio do tempo ou da frequência (LOIZOU, 2013).

FILTRO DE WIENER NO DOMÍNIO DO TEMPO

Neste caso a lógica do filtro é mostrada na figura 12. O sinal desejado é o sinal de voz limpo (*y*[n]*)* o qual é somado ao ruído (*n*[n]*)*; juntos representam o sinal de voz ruidoso *(x*[n]*)* que será filtrado. O filtro têm seus coeficientes ajustados por um algoritmo de adaptação que tenta minimizar a diferença do ruído de referência com o ruído ideal. Portanto, nesse tipo de configuração o próprio erro gerado na saída é também considerado o sinal de voz limpo estimado.

Figura 12 - Diagrama em blocos do Algoritmo Filtro de Wiener do domínio do tempo.



Fonte: Adaptado de Orfanidis (1988, p. 299).

A referência sinal ruído presente no diagrama pode ser obtida em tempos de silêncio (como já mencionado). Assim, assumindo o filtro como de resposta impulsiva finita (FIR – *Finite Inpulse Response*) é possível calcular sua saída com a seguinte expressão.

$$\hat{n}_{ap}[n] = \sum_{k=0}^{M-1} h_k\,[n].\, n_{rf}[n-k] \qquad n = 0,1,2,\ldots \tag{16}$$

Onde: k - Número de amostras do sinal de referência (ruído).

M - Número de coeficientes do filtro.

$\hat{n}_{ap}$ – Ruído calculado, correlacionado com o ruído de referência.

Os coeficientes deste filtro adaptativo são atualizados com algoritmos como o de Widrow-Hoff ou LMS (*Least Mean Squares*) a fim de descorrelacionar o sinal de erro com o sinal de referência. A equação [17] é uma das formas de fazer esta atualização (a teoria sobre os algoritmos de adaptação mencionados pode ser encontrada em ORFANIDIS, 1988).

$$h_k[n+1] = h_k[n] + 2\mu . e[n] . n_{rf}[n] \tag{17}$$

Onde: μ – Escalar para mudar a velocidade de convergência (fixado empiricamente).

$n_{rf}[\mathrm{n}]$ – Amostras do sinal ruído estimado em tempos de silêncio.

Sendo assim, quanto maior for a semelhança do ruído calculado com o real e quanto mais rápido for a convergência ao erro mínimo, melhor será a filtragem do sinal de voz ruidoso.

FILTRO DE WIENER NO DOMÍNIO DA FREQUÊNCIA

A definição do filtro de Wiener no domínio da frequência tem os mesmos princípios anteriormente discutidos, ou seja, a equação que relaciona o erro pode ser representada tanto no domínio do tempo quanto da frequência, conforme as equações [18] e [19]:

$$e[n] = y[n] - \hat{y}[n] \tag{18}$$

$$E[\omega] = \left[Y[\omega] - \hat{Y}[\omega]\right] \tag{19}$$

Onde: $E[\omega]$ e $e[\mathrm{n}]$ – Respectivamente, o espectro e sinal de erro.

$Y[\omega]$ e $y[\mathrm{n}]$ – Respectivamente, o espectro e sinal desejado.

$Y[\omega]$ e $\hat{y}[\mathrm{n}]$ – Respectivamente, o espectro e sinal desejado estimado.

Sabendo que a estimativa do sinal desejado $\hat{y}[\mathrm{n}]$ é gerada por um filtro conforme equação [20], é possível transformar os termos da mesma para o mundo da frequência, resultando na equação [21]:

$$\hat{y}[n] = h[n] * x[n] \tag{20}$$

$$\hat{Y}[\omega] = H[\omega].X[\omega] \tag{21}$$

Onde: $\hat{Y}[\omega]$ – Estimativa do espectro do sinal de voz.

$X[\omega]$ *e* $x[n]$ – Respectivamente, espectro e sinal de voz ruidoso.

$H[\omega]$ *e* $h[n]$ – Respectivamente, função de transferência e coeficientes do filtro.

Definindo $H[\omega]$ de forma a garantir a minimização do MSE, temos que a definição da função de transferência para redução de ruído fica conforme a equação [22]:

$$H[\omega] = \frac{Py}{Py + Pn} \tag{22}$$

Onde: Py – Densidade de potência do espectro do sinal de voz limpo.

Pn – Densidade de potência do espectro do ruído estimado.

A potência do espectro do ruído pode ser estimada através do sistema VAD e/ou assumindo o tempo inicial de áudio como sendo o tempo de silêncio e fazendo uma média dos espectros nesse tempo e calculando sua densidade de potência, conforme equações [13] e [14] anteriormente vistas. Enquanto à potência do espectro do sinal de voz estimado, é um termo desconhecido devido que o mesmo deve ser encontrado posteriormente a equação (equação não causal). Sabendo disso, existem métodos estatísticos e iterativos para tornar possível o cálculo. Além disso, existe uma técnica chamada de decisão-direcionada (DD - *Decision-Directed*) que encontra a SNR a posteriori e a SNR a priori (que são respectivamente, a relação de ruído do sinal de voz ruidoso e do sinal de voz filtrado) e calcula a função de transferência conforme equação a seguir (SCALART, 1996).

$$H[\omega] = \frac{SNR_{priori}}{SNR_{priori} + 1} \tag{23}$$

Mais detalhes sobre esse algoritmo podem ser encontrados em (LOIZOU, 2013) e (SCALART, 1996).

3.3. ERRO QUADRÁTICO MÉDIO MÍNIMO (MMSE)

O algoritmo MMSE é baseado no erro da magnitude dos espectros e não no erro complexo da diferença dos espectros, conforme pode ser visto na equação [24] (LOIZOU, 2013).

$$E[\omega] = \left[|Y[\omega]| - \left|\hat{Y}[\omega]\right|\right] \tag{24}$$

Os algoritmos discutidos até o momento podem ser calculados através da função de transferência *H*[ω] (o filtro de Wiener como mostrado na equação [23] e o de subtração espectral como uma forma parametrizada do filtro de Wiener conforme (LIM; OPPENHEIM, 1979)). A função de transferência do MMSE também pode ser encontrada em função da SNR a priori e a posteriori. A equação [25] mostra a expressão usada em (SCALART, 1996).

$$H[\omega] = \frac{\sqrt{\pi}}{2}\sqrt{\frac{1}{SNR_{post}}\frac{SNR_{priori}}{SNR_{priori}+1}}.F\left[SNR_{post}\left(\frac{SNR_{priori}}{SNR_{priori}+1}\right)\right] \tag{25}$$

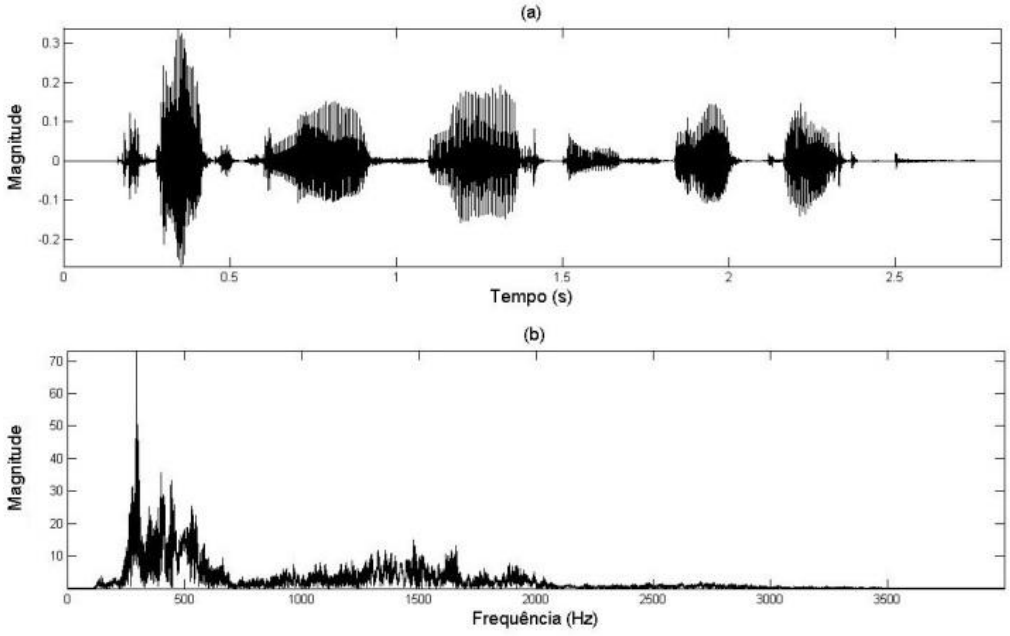
4. SIMULAÇÔES

Os três algoritmos mencionados foram simulados no MATLAB (MATLAB, 2016) utilizando rotinas desenvolvidas por Esfandiar Zavarehei da comunidade MATLAB. As rotinas foram desenvolvidas baseados em artigos publicados em jornais científicos: Subtração Espectral – “SSBoll79.m” (BOLL, 1979), Filtro de Wiener - “WienerScalart96.m” (SCALART, 1996) e Erro Quadrático Mínimo Médio - “MMSESTSA84.m” (EPHRAIM; MALAH, 1984).

Os parâmetros adotados nas simulações foram: análise do sinal usando janelas de Hamm de 25 [ms], sobreposição de 40%, e assunção que os 250 [ms] iniciais do áudio não contém fala, mas apenas ruído. Os sinais de fala foram amostrados a 8 [kHz], ou seja 8 [amostras/ms] o que implica 200 [amostras/janela].

Os sinais de voz e ruídos foram selecionados do banco de dados NOIZEUS Corpus (HU, Yi; LOIZOU, 2007), escolhendo-se um sinal de fala e quatro ruídos ambientes diferentes: ruído captado de dentro de um carro em movimento, ruído captado de dentro de um trem em movimento, ruído ambiente de um restaurante e ruído de rua. Estes ruídos foram adicionados ao sinal de voz com uma SNR de 5 [dB]. Os gráficos do sinal de voz (forma de onda e espectro) escolhido podem ser vistos na figura 13.

Figura 13 - Gráficos do (a) Sinal de Voz no tempo; (b) Espectro do Sinal de Voz.

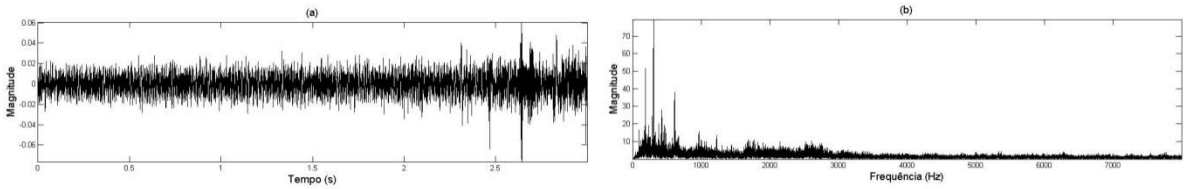


Fonte: própria.

4.1. SINAL DE VOZ COM RUÍDO DE CARRO

O primeiro sinal ruidoso testado é o obtido somando o sinal de voz padrão com o ruído captado de um carro. A figura 14 mostra as características temporais e espectrais desse ruído.

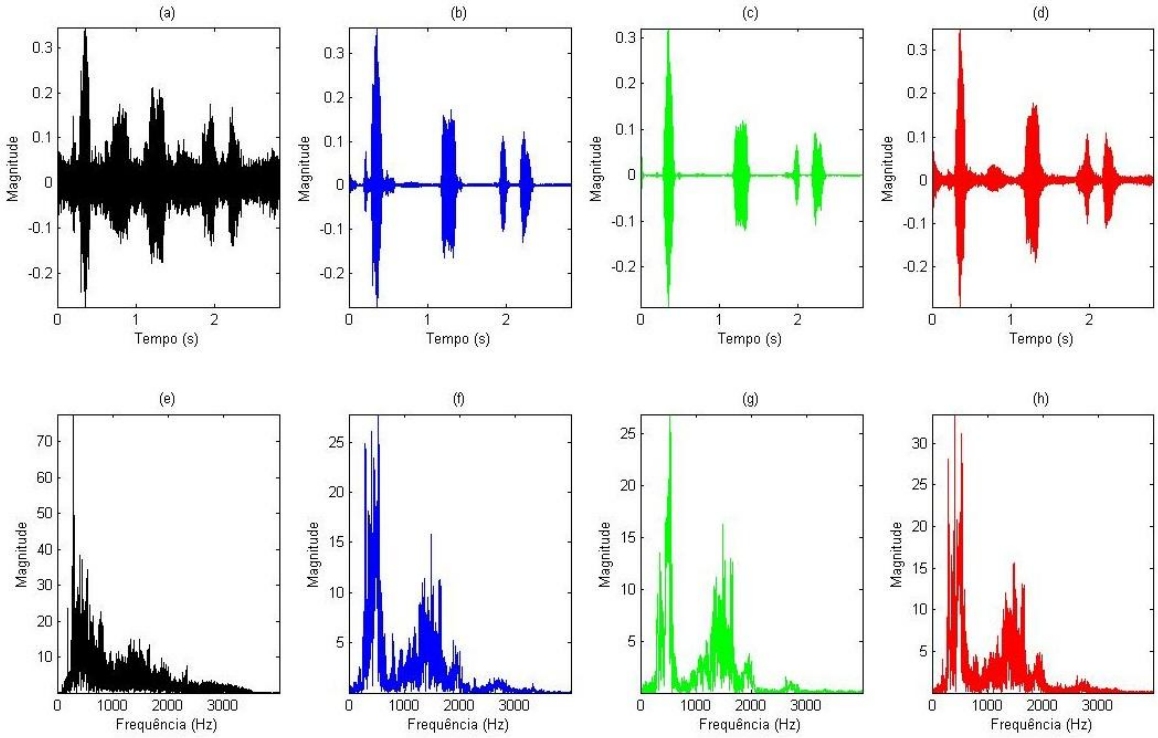
Figura 14 - Gráficos do (a) Ruído de carro no tempo, (b) Espectro do Ruído do carro.



Fonte: própria.

Em seguida, é feita a filtragem do sinal com cada algoritmo analisado. Os resultados da filtragem do sinal de voz com ruído de carro podem ser vistos na figura 15.

Figura 15 - Gráficos do sinal no tempo e na frequência: (a) e (e) sinal de voz com ruído de carro, (b) e (f) Filtrado por Subtração Espectral, (c) e (g) Filtrado pelo filtro de Wiener, (d) e (h) Filtrado por MMSE-STSA.

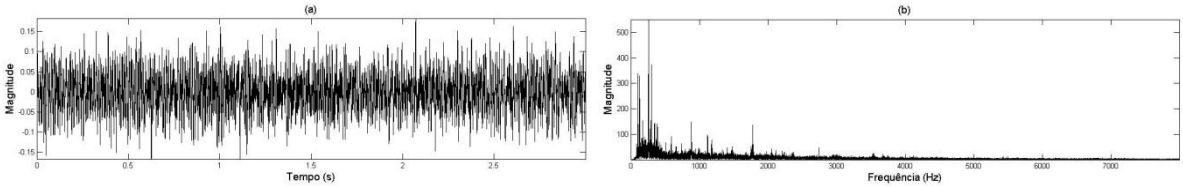


Fonte: própria.

4.2. SINAL DE VOZ COM RUÍDO DE TREM

O segundo sinal de voz ruidoso testado é o obtido somando o sinal de voz com o ruído ambiente captado de um trem. A figura 16 mostra as características temporais e espectrais desse ruído.

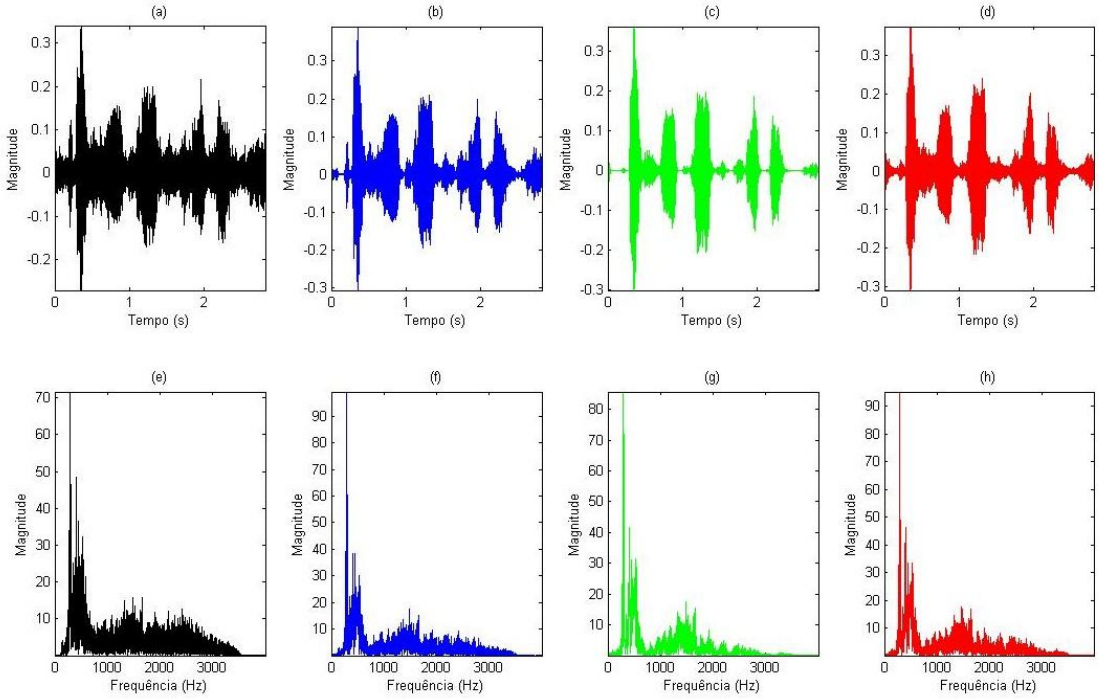
Figura 16 - Gráficos do (a) Ruído de trem no tempo, (b) Espectro do Ruído do trem.



Fonte: própria.

Em seguida, é feita a filtragem do sinal com cada algoritmo analisado. Os resultados da filtragem do sinal de voz com ruído de carro podem ser vistos na figura 17.

Figura 17 - Gráficos do sinal no tempo e na frequência: (a) e (e) sinal de voz com ruído de trem, (b) e (f) Filtrado por Subtração Espectral, (c) e (g) Filtrado pelo filtro de Wiener, (d) e (h) Filtrado por MMSE-STSA.

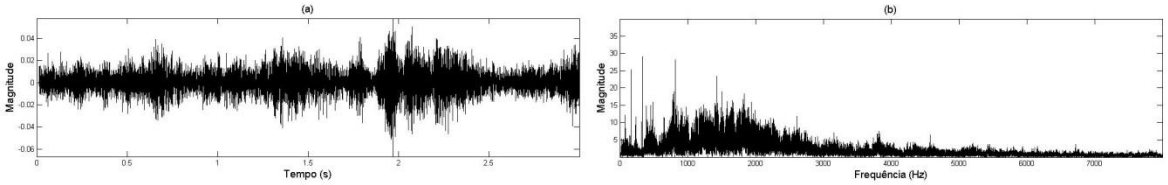


Fonte: própria.

4.3. SINAL DE VOZ COM RUÍDO AMBIENTE DE UM RESTAURANTE

O terceiro sinal de voz ruidoso testado é obtido somando o sinal de voz ao ruído ambiente captado de dentro de um restaurante. A figura 18 mostra as características temporais e espectrais desse ruído.

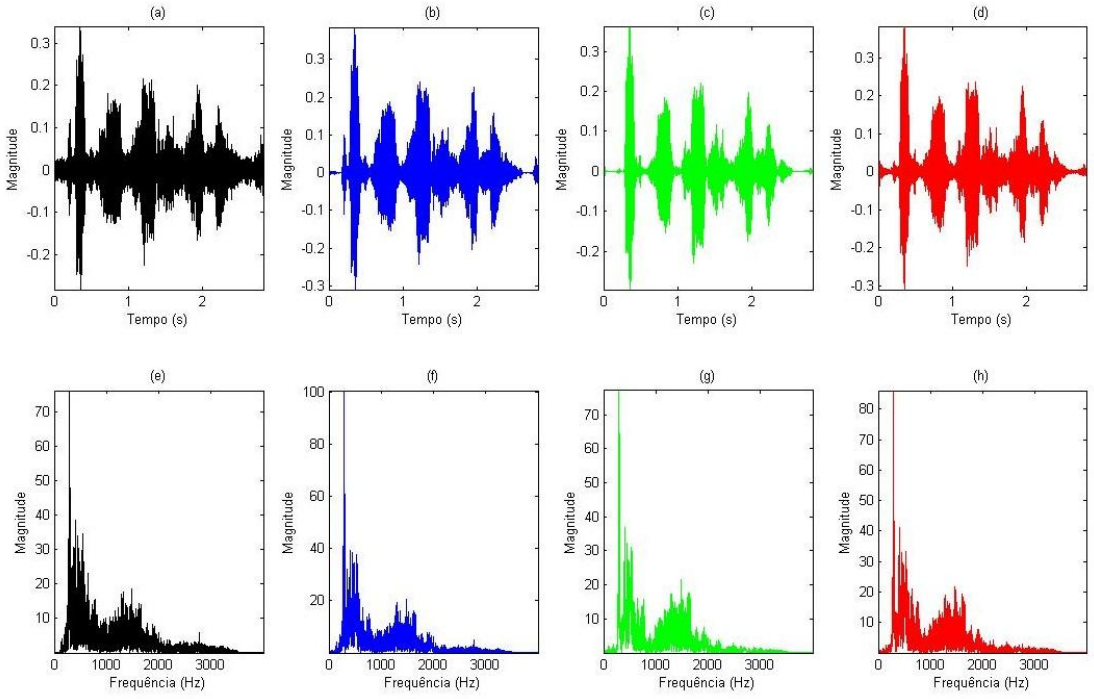
Figura 18 - Gráficos do (a) Ruído do restaurante no tempo, (b) Espectro do Ruído do restaurante.



Fonte: própria.

Em seguida, é feita a filtragem do sinal com cada algoritmo analisado. Os resultados da filtragem do sinal de voz com ruído de carro podem ser vistos na figura 19.

Figura 19 - Gráficos do sinal no tempo e na frequência: (a) e (e) sinal de voz com ruído do restaurante, (b) e (f) Filtrado por Subtração Espectral, (c) e (g) Filtrado pelo filtro de Wiener, (d) e (h) Filtrado por MMSE-STSA.

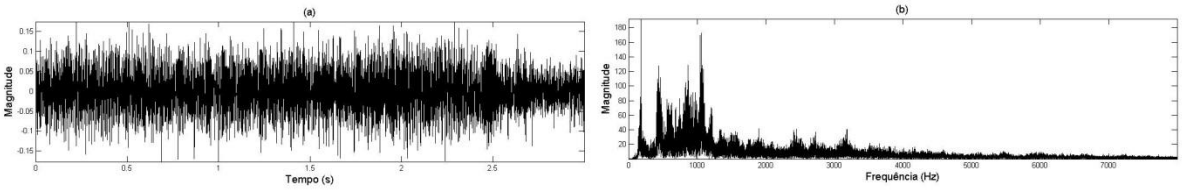


Fonte: própria.

4.4. SINAL DE VOZ COM RUÍDO AMBIENTE DA RUA

O quarto sinal de voz ruidoso testado é o obtido somando o sinal de voz padrão com o ruído ambiente captado de uma rua pública. A figura 20 mostra as características temporais e espectrais desse ruído.

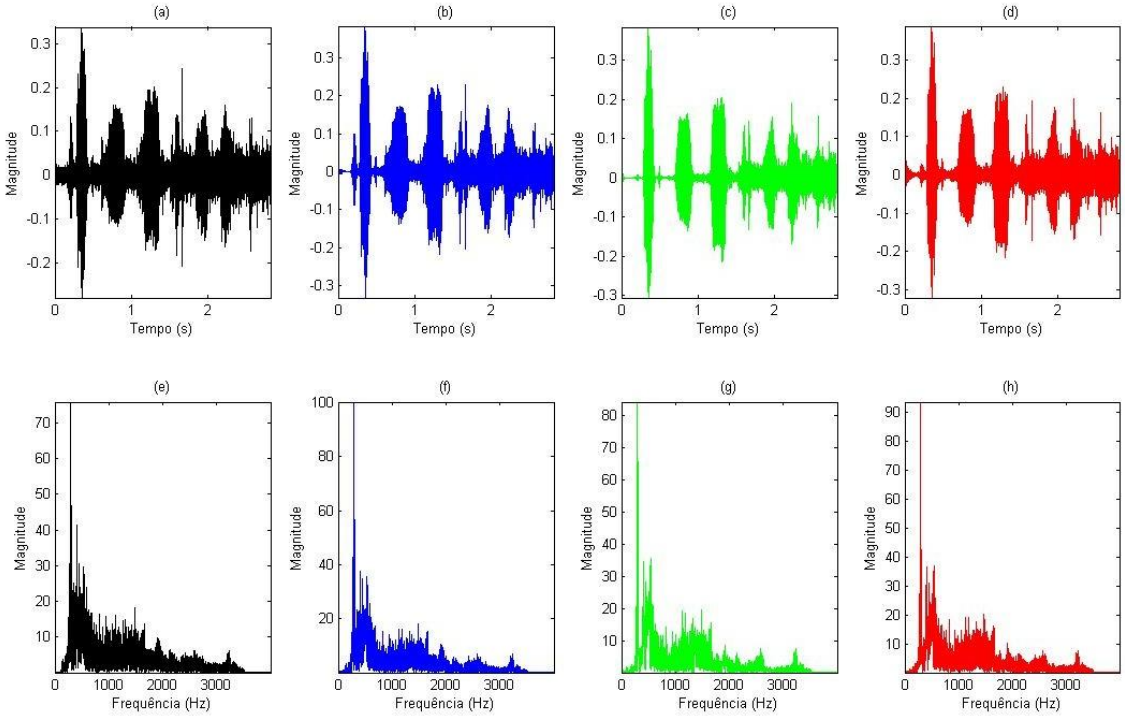
Figura 20 - Gráficos do (a) Ruído da rua no tempo, (b) Espectro do Ruído da rua.



Fonte: própria.

Em seguida, é feita a filtragem do sinal com cada algoritmo analisado. Os resultados da filtragem do sinal de voz com ruído de carro podem ser vistos na figura 21.

Figura 21 - Gráficos do sinal no tempo e na frequência: (a) e (e) sinal de voz com ruído de rua, (b) e (f) Filtrado por Subtração Espectral, (c) e (g) Filtrado pelo filtro de Wiener, (d) e (h) Filtrado por MMSE-STSA.



Fonte: própria.

4.5. RESULTADOS DA SIMULAÇÃO

Para analisar a eficiência dos algoritmos é realizado o cálculo de SNR dos quatro sinais de voz filtrados, através da equação [1.a] e [1.b], e o resultado desses testes é visto na tabela a seguir.

Tabela 3 - Resultados encontrados no cálculo da relação sinal-ruído (em decibéis).

SNR (dB)	Carro	Trem	Restaurante	Rua
Subtração Espectral	8,58	10,23	8,42	10,73
Filtro de Wiener	15,71	12,62	8,63	17,35
MMSE	6,14	9,10	8,10	7,61

Fonte: própria.

Analisando os resultados é visto que a relação SNR anteriormente de 5 dB passa a ser maior depois da filtragem, ou seja, a influência do ruído está menor, provando a eficiência dos algoritmos nesse teste objetivo. Outro tipo de teste objetivo é a SSNR antes mencionada (equação [2]), mas devido a problemas com o *script* nativo dessa função no MATLAB, esse teste não apresentou resultados coerentes.

5. IMPLEMENTAÇÃO DO ALGORITMO NO DSP

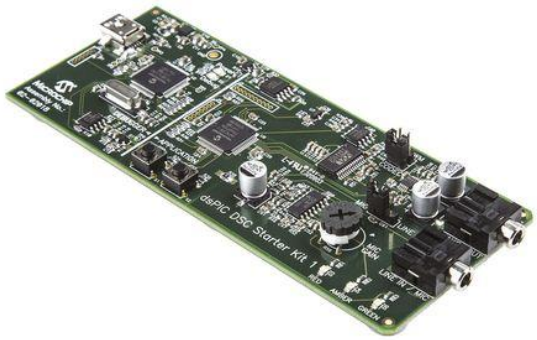
5.1. HARDWARE E INFRAESTRUTURA

A plataforma escolhida para implementação do algoritmo de redução de ruído é a **MPLAB Starter Kit for dsPIC DSC** da empresa Microchip (MICROCHIP, 2016). Foi escolhida depois de um estudo das placas com DSP disponíveis no mercado destinadas a processamento de áudio. Nesse estudo foram levadas em conta as características da placa, seu preço e a facilidade de aquisição.

Levado em consideração para escolha da placa, a familiaridade do autor deste trabalho com outros processadores da empresa, suporte da comunidade que aborta ideias e soluções *online*, as bibliotecas disponíveis, o acesso grátis ao ambiente de desenvolvimento integrado MPLAB X IDE (*Integrated Development Environment*) e ao compilador de linguagem C MPLAB XC16 (MICROCHIP, 2016).

A figura 22 mostra o aspecto físico da placa; suas dimensões são 120 x 50 milímetros, ou seja, é de tamanho reduzido. Sua alimentação é de +5 [V] através de um conector mini USB (*Universal Serial Bus*).

Figura 22 - Placa da plataforma MPLAB *Starter Kit for dsPIC DSC*



Fonte: (MICROCHIP, 2016).

A plataforma permite, através de uma porta USB, a gravação e/ou depuração do código gerado para o processador DSP, ou seja, não há necessidade de outro dispositivo para essas funções.

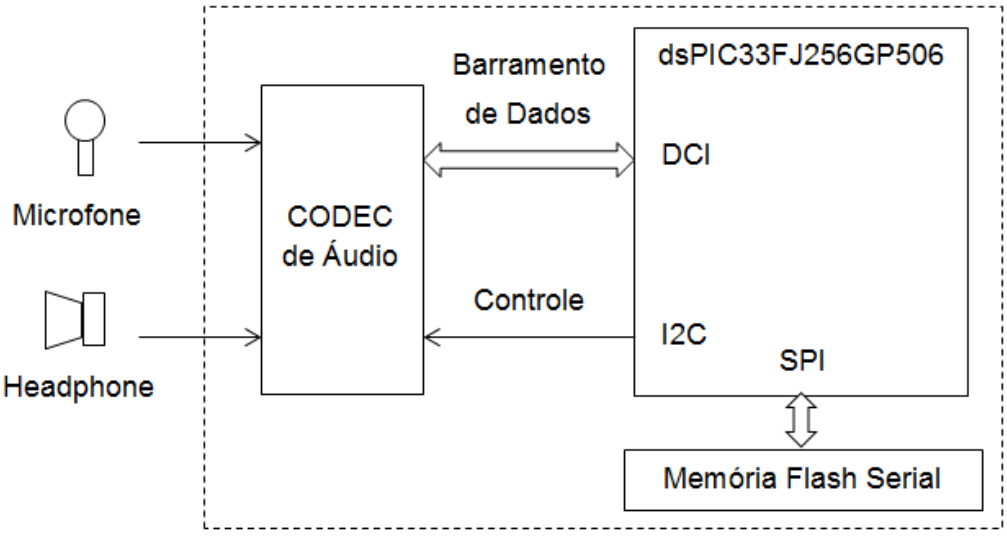
O DSP incorporado na placa é o dsPIC33FJ256GP506 da família dsPIC33F da Microchip e é um DSP de ponto-fixo de 16 bits. Contém 256 [kB] de memória de programa e 16 [kB] de memória RAM (*Random Access Memory*). Este DSP é considerado também um DSC (*Digital Signal Controller*), pois contém características

de processamento de um DSP com a adição de alguns componentes comumente encontrados em microcontroladores.

Por se tratar de uma placa para processamento de áudio, a plataforma contém um CODEC (acrônimo de codificador/decodificador) com conversor A/D com resolução de até 24 bits e frequência máxima de amostragem de 48 [kHz], uma entrada para microfone e uma saída para fone de ouvido com impedância de 32[Ω]. O kit também contém uma memória flash externa com comunicação serial do tipo SPI (*Serial Peripheral Interface*) e capacidade de 4 [Mb] (512 [kB]) que pode ser utilizada para armazenar qualquer tipo de dado, como por exemplo, um sinal de áudio capturado pelo microfone.

O processador se comunica com o CODEC através de um barramento de dados do tipo DCI (*Data Converter Interface*) e outro de controle do tipo serial I^2C (*Inter-Integrated Circuit*). Através do barramento de controle é possível ordenar a transferência de dados e configurar alguns parâmetros do CODEC como, por exemplo, o protocolo de transferência de dados que será utilizado, a frequência de amostragem desejada, a resolução do conversor A/D, o volume do sinal de saída, a filtragem do sinal de entrada etc. A figura 23 mostra o diagrama de blocos simplificado da placa para aplicações com áudio.

Figura 23 - Diagrama da Plataforma MPLAB Starter Kit for dsPIC.

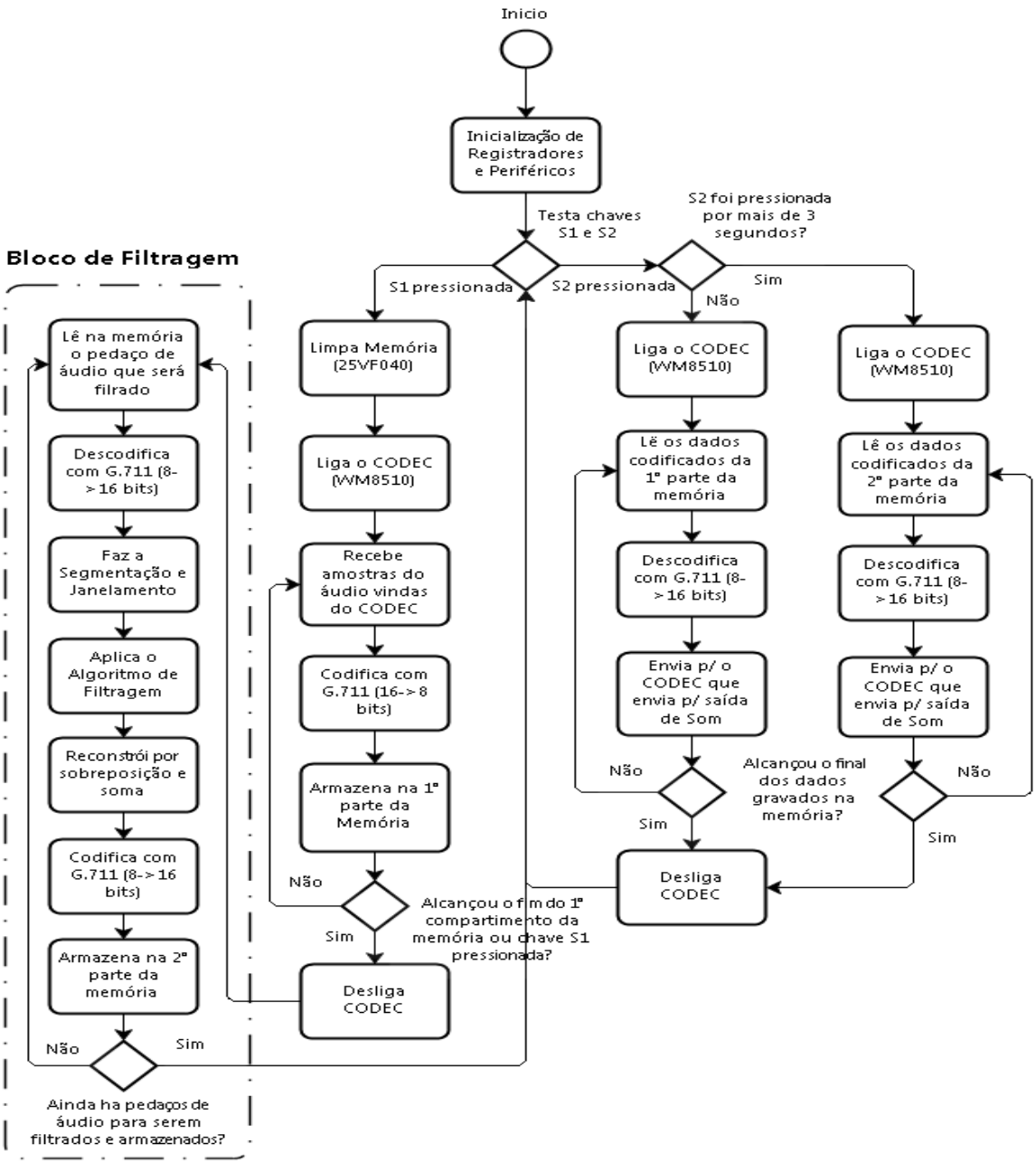


Fonte: Adaptado de MPLAB Starter Kit for dsPIC (2016, p.14).

5.2. VISÃO GERAL DO SISTEMA

Na figura 24 é apresentado um diagrama de fluxo da solução, que começa pela inicialização dos registradores e periféricos. Em seguida o programa fica em *loop* até que alguma das chaves seja pressionada; caso seja a chave S1, ocorre o processo de gravação e filtragem do áudio; caso seja a S2, ocorre a reprodução do áudio original, caso a mesma seja pressionada por mais de 3 segundos, ocorre à reprodução do áudio filtrado.

Figura 24 - Representação do sistema através de um fluxograma geral do código.



Fonte: própria.

5.3. GRAVAÇÃO E REPRODUÇÃO

De acordo com a escolha da aplicação de áudio é possível definir alguns valores de frequência de amostragem e resolução das amostras para o conversor A/D ou CODEC. A Tabela 4 mostra algumas combinações interessantes de acordo com a aplicação.

Tabela 4 – Valores digitais de processamento vs. aplicações de áudio.

Tipo de Aplicação	Freq. de Amostragem (kHz)	Bits por Amostra	Número de Canais	Taxa de Transferência (kB/s)	Banda de Frequência (kHz)
Telefonia	8	8	Mono	8	0,2 – 3,4
Rádio AM	11,025	8	Mono	11	0,1 – 5,5
Rádio FM	22,05	16	Estéreo	88,2	0,02 – 11
CD	44,1	16	Estéreo	176,4	0,005 – 20
DVD Áudio	Até 192	Até 24	6 canais	Até 1.200,0	0 – até 96

Fonte: Adaptado de Li e Drew (2004, p.137).

Os valores usados em telefonia, de qualidade aceitável para uso comercial, são de 8 [kHz] para a frequência de amostragem e 8 bits/amostra para a quantização. Neste trabalho, como o foco é a filtragem de ruídos em sinais de voz (conversação), será utilizada uma frequência de amostragem de **8 [kHz]** e quantização com **16 bits/amostra**. Esse aumento da resolução das amostras implica em uma melhor aproximação dos valores analógicos e com isso há uma melhoria no áudio, mas também implica em um aumento da taxa de transferência, o que exige do DSP um maior poder de processamento e maior memória ou capacidade de armazenamento. Define-se como taxa transferência do conversor A/D, o produto da frequência de amostragem pelo número de bits de quantização; sendo assim, temos que neste trabalho a taxa de transferência foi de:

$$Taxa\ de\ Transf. = (8kHz)x(16b/am) = \mathbf{128}\ [\boldsymbol{kb/s}]\ ou\ \mathbf{16}\ [\boldsymbol{kB/s}] \qquad (26)$$

Outra consequência desse aumento na resolução é a memória necessária para armazenar os dados de áudio amostrados. Para calcular o tamanho de memória necessária para armazenar um sinal de voz de duração T[s], usa-se a equação [27].

$$Tamanho\ \ [B] = Taxa\ de\ Transferência\ [B/s]\ x\ \ T[s] \quad (27)$$

Sabendo que a memória RAM do DSP é de 16 [kB], só seria possível armazenar:

$$Duração\ do\ Áudio = \frac{Tamanho\ da\ Memória\ [B]}{Taxa\ de\ Transferência\ [B/s]} = \frac{16\ [kB]}{16\ [kB/s]} = \mathbf{1}\ [\boldsymbol{s}] \quad (28)$$

Seria penoso fazer uma aplicação embarcada sem o uso de uma memória externa. Como dito anteriormente, na placa escolhida há uma memória *flash* externa de 512 [kB] de capacidade, sendo assim:

$$Duração\ do\ Áudio = \frac{Tamanho\ da\ Memória\ [B]}{Taxa\ de\ Transferência\ [B/s]} = \frac{512\ [kB]}{16\ [kB/s]} = \mathbf{32}\ [\boldsymbol{s}] \quad (29)$$

Contudo, existem técnicas de codificação e compressão para diminuir a taxa de transferência e consequentemente a memória necessária para armazenamento do sinal de áudio.

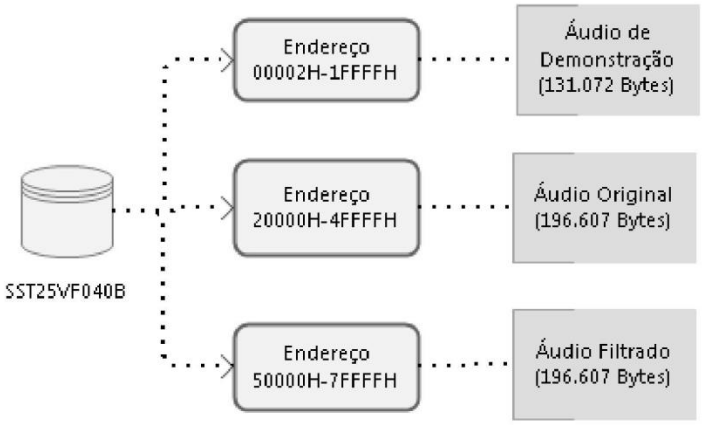
A codificação utilizada nesse trabalho é denominada de G.711 (lei-μ). Sua origem vem da telefonia onde os sinais podem ser codificados a uma taxa de 64 [kb/s], ou seja, 8 [kHz] e 8 [b/am]. Este codificador mapeia de 16 bits para 8 bits as amostras quantizadas utilizando uma lei logarítmica, tornando a taxa de transferência de 8 [kB/s] (G.711, 1972).

Os dados de áudio codificados são armazenados no bloco de endereço 20000H - 4FFFFH (hexadecimal), devido ao primeiro bloco de memória 2H - 1FFFFH ser ocupado por um áudio de demonstração nativo da placa, o qual foi preservado para ser utilizado caso ocorra algum problema com um dos periféricos. Com isso, temos 192 [kB] de espaço para armazenar o áudio codificado a uma taxa de transferência de 8kB/s. Assim, podem ser armazenados até 24[s] de áudio:

$$T[s] = \frac{Tamanho\ [B]}{Taxa\ de\ Transferência\ [B/s]\ x} = \frac{192\ [kB]}{8\ [kB/s]} \cong \mathbf{24}[\boldsymbol{s}] \quad (30)$$

Quando a escrita do áudio na memória chega ao endereço final, o algoritmo finaliza a captura evitando assim a invasão de outro bloco de memória. O terceiro bloco da memória é utilizado para armazenar o áudio filtrando, o qual tem o mesmo tamanho do áudio original. A figura 25 mostra como fica a divisão de memória.

Figura 25 - Diagrama da divisão do áudio na memória flash.



Fonte: (própria).

Vale descrever que no processo de gravação os dados de áudio capturados pelo CODEC são armazenados em um *buffer* temporário de dados de 128 amostras, o qual deve ser lido antes das próximas 128 amostras chegar; isto dá um intervalo de tempo inversamente proporcional a frequência de amostragem para serem feitas as operações de codificação e armazenamento dos dados. O intervalo de tempo proporcionado é dado pela equação 31.

$$Int.Tempo = \frac{Tam.do\ Buffer}{Freq.Amost.} = \frac{128\ [ams]}{8000\ [ams/s]} = \mathbf{0,016}[\boldsymbol{s}]\ ou\ \mathbf{16}[\boldsymbol{ms}] \qquad (31)$$

O processo de reprodução é inverso ao de gravação; começa com a leitura da memória, descodificação e envio ao *buffer* de reprodução. Considerando que o procedimento seja em tempo real, as 128 amostras devem ser lidas da memória e descodificadas antes que tenha acabado a reprodução das últimas amostras.

5.4. ALGORITMO DE FILTRAGEM

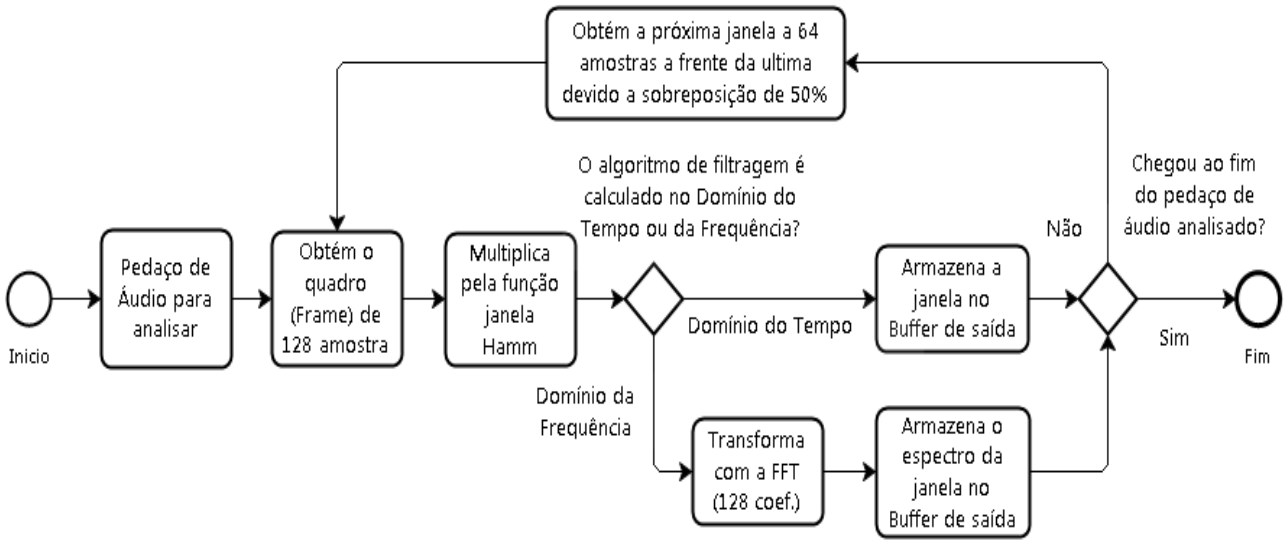
5.4.1. **SEGMENTAÇÃO DO ÁUDIO**

Conforme a explicação feita sobre análise de sinais de áudio, o algoritmo deve executar a segmentação do sinal de áudio em quadros ou janelas de 25[ms]. O mais apropriado para a frequência de amostragem de 8 [kHz] é um quadro de 16[ms], pois, o mesmo proporciona um número de amostras que é potência de dois (equação [32]) que é um requisito para a FFT.

$$Tamanho\ da\ Janela\ = (Duração\ da\ Janela)x\ (Freq.de\ Amostragem)$$
$$Tamanho\ da\ Janela = \ 0{,}016\ [s]\ x\ 8000\ [ams/s] = \mathbf{128}\ [\boldsymbol{ams}] \quad (32)$$

Adotando uma sobreposição de 50%, a cada 64 amostras deve analisar-se um quadro de 128 amostras. As amostras capturadas são então ponderadas com a janela de Hamm (equação [6]). Se o algoritmo de filtragem trabalha no domínio do tempo, basta armazenar os valores das amostras ponderadas no *buffer* de saída, se não, é feita a FFT da janela (com no mínimo 128 coeficientes) e armazenadas os valores da FFT (valores complexos) no *buffer* de saída. O fluxograma abaixo proporciona uma visão deste processo de análise.

Figura 26 - Fluxograma do algoritmo de análise.



Fonte: própria.

Na prática, o intervalo de áudio analisado por vez é de tamanho limitado devido ao tamanho reduzido da memória RAM do DSP; assim, um intervalo de 512 amostras (64 [ms]) pode ocupar um grande espaço. O cálculo do consumo de memória RAM utilizada no algoritmo inicia-se pelo cálculo do número de janelas que serão geradas com o intervalo de áudio sob análise, conforme a equação [33]:

$$N° \; de \; Janelas = \left[\left(\frac{Tam. áudio \; em \; Análise - Tamanho \; da \; Janela}{Passo \; de \; Análise}\right) + 1\right]$$
$$N° \; de \; Janelas = \left[\left(\frac{512 \; [ams] - 128 \; [ams]}{64 \; [ams]}\right) + 1\right] = \mathbf{7} \; [\boldsymbol{janelas}] \quad (33)$$

Tendo o número de janelas é possível calcular o tamanho do *buffer* de saída necessário para armazenar as janelas após a segmentação, seja no domínio do tempo ou no domínio da frequência (nesse domínio, é necessário o dobro do tamanho devido às componentes resultantes da FFT serem complexas). As equações [34] e [35] mostram este cálculo:

$$Buffer \; no \; D.T = (Tam. da \; Janela) \; x \; (N° \; de \; Janelas) \; x \; (Tam. da \; Amostra)$$
$$Buffer \; no \; D.T \; (Saída) = 128 \; [ams] \; x \; 7 \; [janelas] \; x \; 16 \; [b] \quad (34)$$
$$Buffer \; no \; D.T \; (Saída) = \mathbf{14.336} \; [\boldsymbol{b}] \; ou \; \mathbf{1}\,,\mathbf{75}[\boldsymbol{kB}]$$

$$Buffer \; no \; D.F = (Tam. da \; Janela) \; x \; \; (N° \; de \; Janelas) \; x \; (Tam. da \; Comp.)$$
$$Buffer \; no \; D.F \; (Saída) = 128 \; [ams] \; x \; 7 \; [janelas] \; x \; 32 [b] \quad (35)$$
$$Buffer \; no \; D.F \; (Saída) = \mathbf{28.672} \; [\boldsymbol{b}] \; ou \; \mathbf{3}\,,\mathbf{5}[\boldsymbol{kB}]$$

Temos então que o valor em bytes para analisar um intervalo de áudio de 512 amostras é de:

$$Cons. de \; Memória \; (D.T) = Buffer \; (Entrada) + Buffer \; no \; D.T \; (Saída)$$
$$Cons. de \; Memória \; (D.T) = 1[kB] + 1{,}75 \; [kB] = \mathbf{2,75}\boldsymbol{kB} \quad (36)$$

$$Cons. de \; Memória \; (D.F) = Buffer \; (Entrada) + Buffer \; no \; D.F \; (Saída)$$
$$Consumo \; de \; Memória \; (D.T) = 1[kB] + 3{,}5 \; [kB] = \mathbf{4,5}\boldsymbol{kB} \quad (37)$$

Assim, um intervalo de 512 amostras ocupa esse espaço da memória RAM do DSP; o restante de memória RAM é utilizado por outras variáveis do sistema.

5.4.2. ALGORITMO QUE IMPLEMENTA A FILTRAGEM

Uma vez o *buffer* preenchido com as sete janelas analisadas, o conteúdo deve ser filtrado com o algoritmo de redução de ruído. Neste trabalho, foram implementados no DSP: a subtração espectral baseada no artigo da rotina do MATLAB (BOLL, 1979), o filtro de Wiener baseado em (SCALART, 1996) e por último o filtro de Wiener no domínio do tempo com o algoritmo de adaptação LMS baseado em (ORFANIDIS, 1988) e (CHASSAING, 2003).

Em função dos resultados obtidos, os quais serão apresentados na seção 5.5 (RESULTADOS DA IMPLEMENTAÇÃO), nesta seção será explicado, apenas, a implementação em DSP do algoritmo de Subtração Espectral. A figura 27 mostra o fluxograma correspondente.

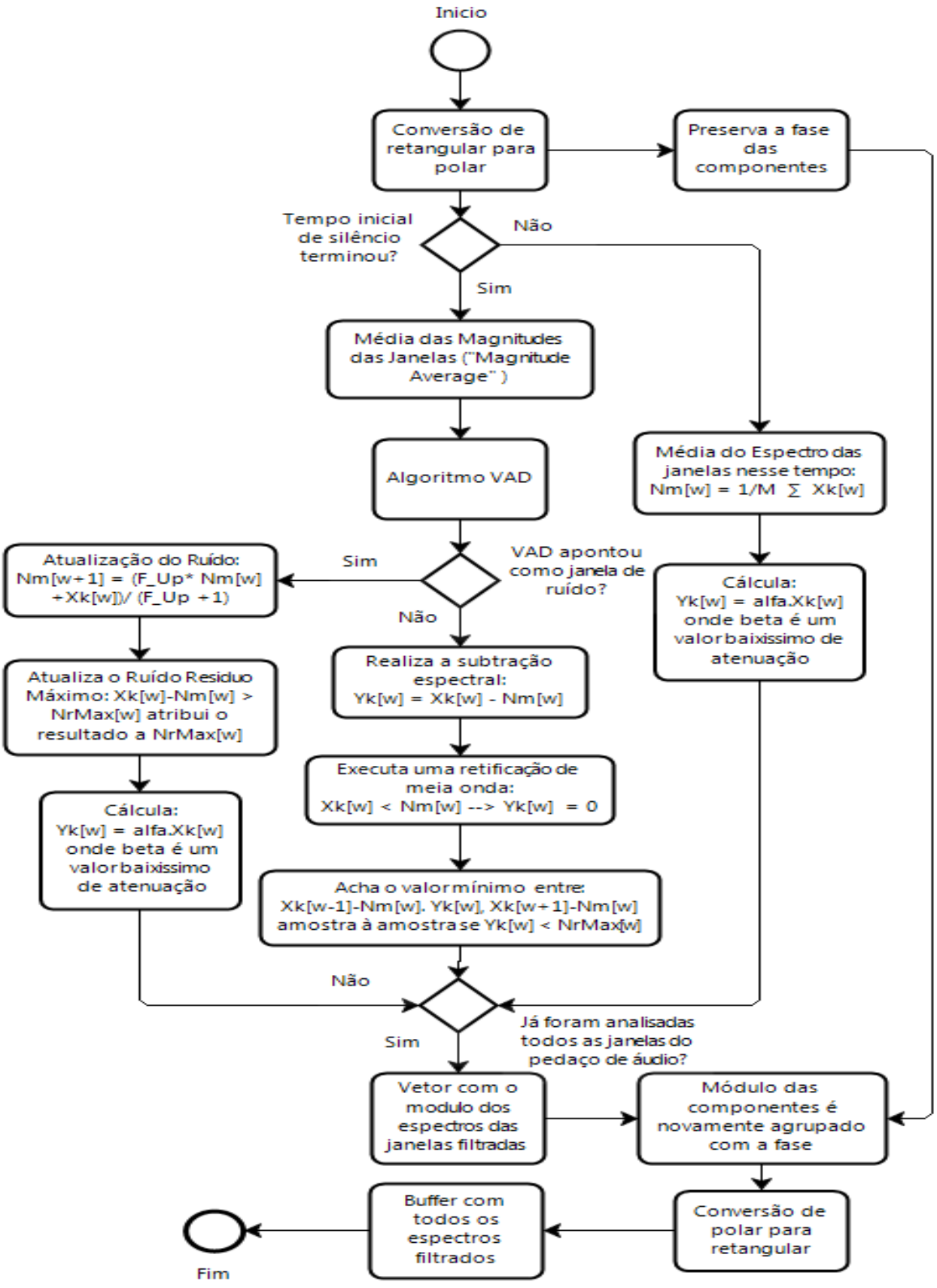
O algoritmo se inicia com a conversão das componentes espectrais obtidas da FFT, que estão na forma retangular (parte real e imaginária), para a forma polar (módulo e fase). Os módulos das componentes é que serão analisados, enquanto que a fase das componentes é preservada e só utilizada na reconstrução do áudio no final do algoritmo.

Calculado o módulo das janelas, se estas correspondem ao início do áudio, que é assumido sem fala, faz-se a estimativa do ruído com a equação [14] (média de todas as janelas nesse intervalo de tempo). Adicionalmente, já que o conteúdo destas janelas é ruído, ele é atenuado utilizando uma constante empírica alfa. O intervalo inicial de silêncio adotado neste trabalho foi de

$$\begin{gathered} Tempo\ de\ Silêncio = 4\ intervalos\ com\ 512\ amostras\ cada \\ Tempo\ de\ Silêncio = 4\ x\ 64\ [ms]\ = \mathbf{256}\ [\boldsymbol{ms}] \end{gathered} \tag{38}$$

Depois de processado o tempo do silêncio inicial, é feita a média de cada componente da janela com as respectivas janelas anterior e posterior. Essa média é chamada em inglês de *Magnitude Average*, e seu propósito é diminuir o ruído residual caso a janela contenha um ruído mais alto que a média espectral do ruído subtraída dela.

Figura 27 - Fluxograma do algoritmo de redução ruído de Subtração Espectral implementado.



Fonte: própria.

O cálculo desta média deve ser feito entre janelas totalizando um máximo de 38,4 [ms] para evitar perder inteligibilidade no sinal de voz. Neste trabalho, dado que as janelas de análise são de 16 [ms] e que há uma sobreposição de 50% entre elas, o tempo utilizado para este cálculo foi de 32 [ms].

Após este passo, o *buffer* contém os módulos médios de cada janela, os que são utilizados em todos os cálculos seguintes.

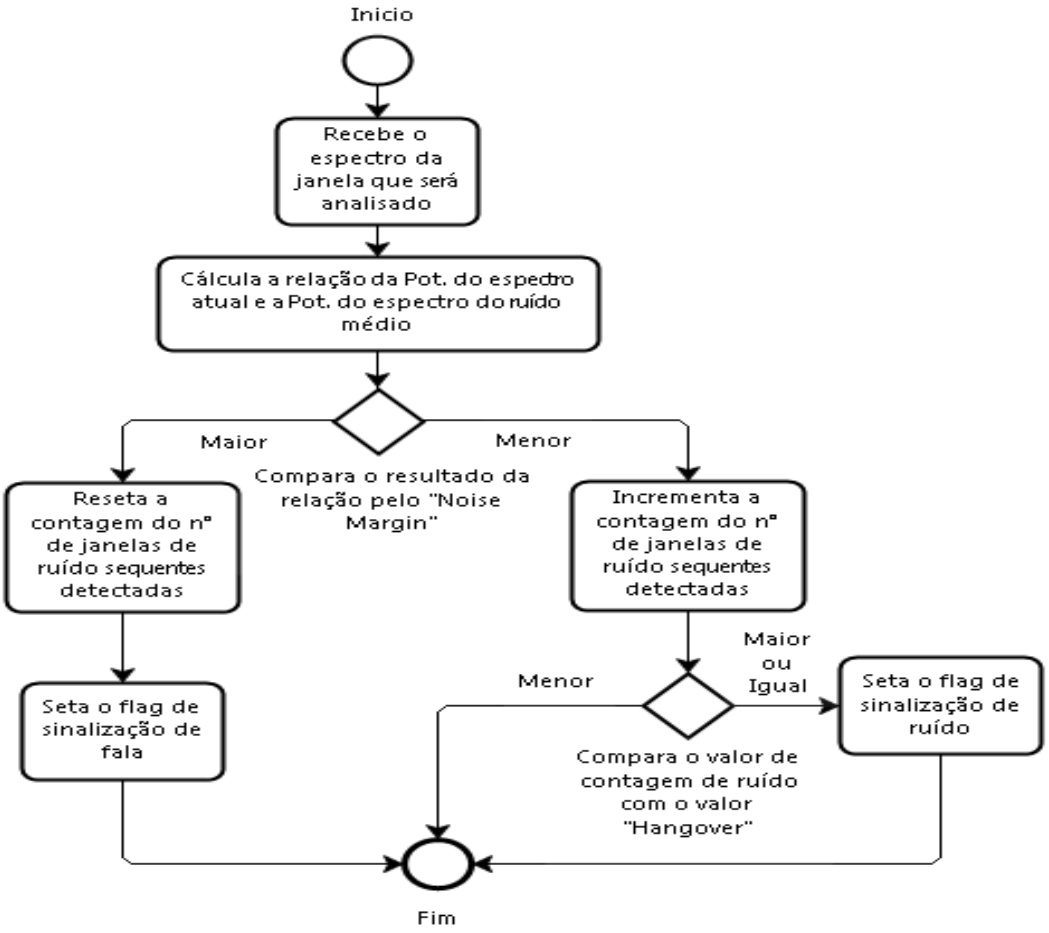
Na próxima etapa do algoritmo todas as janelas são analisadas por um algoritmo VAD que detecta a atividade de voz nas janelas. O algoritmo VAD implementado é uma versão simples mostrada na figura [28]. O mesmo é baseado na relação da potência do espectro da janela analisada com o espectro do ruído médio estimado. Quando há uma diferença grande entre ambas potências (3dB - 50% a mais de potência), ou seja, a potência da janela analisada é bem maior do que a potência do ruído estimado, a janela é considerada como contendo fala; caso a diferença seja baixa (abaixo de 50%), é incrementado um contador de ruídos consecutivos (esse contador evita que alguns sons de baixa potência da fala sejam consideradas como ruído, e foca em sinalizar o ruído estacionário). Para funcionar, o VAD utiliza dois limiares empíricos: o *Noise Margin* (margem de ruído – antes mencionado de 3dB) e *o Hangover* (resíduo), que são, respectivamente, o valor que determina a fronteira entre diferença de potências baixa e alta, e o valor de arrasto do ruído, ou seja, valor que o contador de ruídos tem que alcançar para que as janelas possam ser consideradas como ruído.

Se o VAD define a janela como ruído, ocorre uma atualização do espectro do ruído médio estimado anteriormente. A atualização do ruído pode ser feita com a equação [13] ou com a equação [39].

$$\left|\widehat{N}[\omega]\right|_T = \frac{\left((Factor\ Update)\ x\ \left|\widehat{N}[\omega]\right|_{T-1}\right) + |X[\omega]|_T}{Factor\ Update + 1} \tag{39}$$

onde *Factor Update* (Fator de Atualização) é um valor de consideração, ou seja, o quanto as novas janelas de ruídos são relevantes diante do ruído médio estimado anteriormente. Quanto menor o fator de atualização, mas rápida será a atualização do ruído com as novas características.

Figura 28 - Fluxograma do algoritmo de VAD implementado.



Fonte: própria.

Outro item calculado quando a janela é definida como ruído, é o ruído residual máximo. Conforme mencionado anteriormente, esse resíduo de ruído é encontrado devido à estimativa do ruído ser baseada em uma média. O valor máximo do resíduo é calculado subtraindo da janela atual (considerada de ruído pelo VAD) do ruído médio calculado anteriormente. Caso o resultado dessa subtração seja maior que os valores anteriores de ruído residual máximo, ele é atualizado. Em seguida, assim como é feito nos primeiros 256[ms] de áudio, considerados de silêncio, a janela tem suas componentes atenuadas multiplicando-as pelo valor alfa já mencionado.

Caso a janela seja considerada como fala pelo algoritmo VAD, é feita a subtração espectral (espectro do conteúdo da janela menos o espectro do ruído médio estimado) conforme equação [12]. Em seguida é feita a retificação de meia-onda (conforme equação [15]). Logo ocorre à comparação das componentes resultantes da janela com os valores de ruído residual máximo; caso alguma componente seja menor, compara-se a componente com as correspondentes da

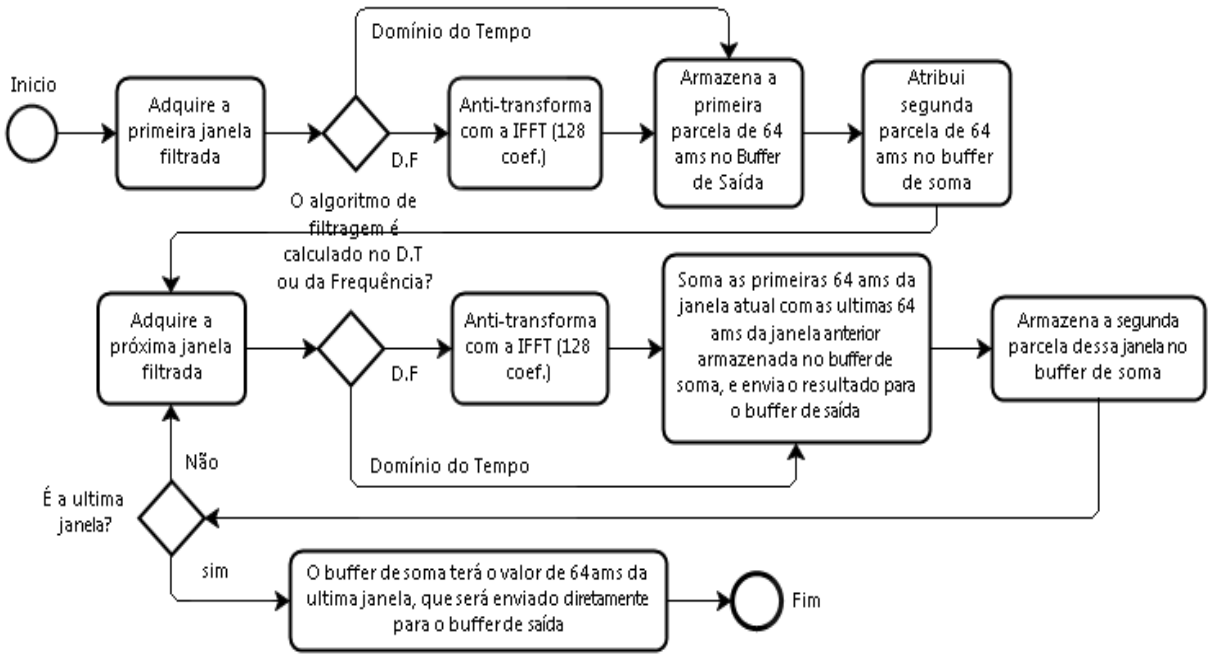
janela anterior e posterior subtraídas do ruído médio, e a menor das três é escolhida como o novo valor da componente.

O processo se repete até esgotar as 7 janelas de áudio do intervalo sob análise, para então repetir todo novamente até esgotar o áudio de entrada.

5.4.3. RECONSTRUÇÃO DO ÁUDIO

Após a análise e filtragem de cada janela, ocorre a reconstrução do intervalo de áudio utilizando a técnica de sobreposição e soma (equação [7]). Nessa técnica é necessário diferenciar a primeira e última janela de analise pois, respectivamente, os 50% iniciais e finais delas não tem sobreposição. O fluxograma abaixo proporciona uma visão melhor do algoritmo de reconstrução do intervalo de áudio, salientando a necessidade de calcular a IFFT (*Inverse Fast Fourier Transform*), em casos de algoritmos como o de Subtração Espectral que trabalha com o espectro do conteúdo das janelas.

Figura 29 - Fluxograma do algoritmo de reconstrução do áudio por sobreposição e soma.



Fonte: própria.

5.5. RESULTADOS DA IMPLEMENTAÇÃO

Em termos de qualidade de filtragem, a implementação com o algoritmo de subtração espectral apresentou os melhores resultados. O mesmo conseguiu a eliminação total do ruído estacionário em tempos em que não há fala, e para os trechos em que há fala, o mesmo deixou um vestígio do ruído no fundo, mas não alterou as características da voz.

O baseado no filtro de Wiener no domínio da frequência apresentou um ruído de fundo musical (*musical noise*) em trechos em que não há fala, e uma mudança da tonalidade da voz do locutor (a voz ficou um pouco mais grave).

A implementação do algoritmo de filtro de Wiener com o algoritmo de convergência LMS apresentou problemas. Basicamente, a adaptação do filtro não trabalhava bem com a faixa dinâmica do sinal de áudio, pois quando o valor de amplitude era baixo o algoritmo não convergia para o estado de erro mínimo, e caso fosse tentado escalonar o valor de entrada, ocorria overflow (saturação) das variáveis de 16 bits do DSP. Sabendo disso, foi feito um estudo de algumas modificações nas variáveis chaves do algoritmo para 32 bits, contudo, o mesmo não apresentou bons resultados para sinais de áudio, deixando o sinal de saída igual ao de entrada, ou seja, sem nenhuma filtragem (para outros sinais, desde que com faixa dinâmica baixa e estacionários, houve convergência).

No aspecto desempenho, as implementações atuais dos algoritmos de Subtração Espectral e de Wiener no domínio da frequência não poderiam ser utilizadas para filtragem em tempo real, já que para a filtragem demoram de 3 a 4 vezes o tempo do áudio original. Já para aplicações onde não é exigido tempo real, as implementações são suficientes.

Uma das razões para o desempenho atual é a quantidade e tipo das operações matemáticas que executa o algoritmo. Nos dois casos houve a necessidade de realizar operações de conversão para o formato polar das componentes espectrais. Essas conversões são do tipo trigonométrico, e constituídas de diversas operações além da multiplicação, soma e acúmulo, que são as operações padrão do DSP (unidade MAC - *Multiplier–Accumulator*, do inglês, Multiplicador-Acumulador).

Outra razão é que o DSP utilizado na implementação é de ponto fixo (16 bits tipo Q15) e há necessidade de converter algumas variáveis para ponto flutuante em

partes do algoritmo. Utilizando um DSP de ponto flutuante (32 bits) não haveria esta necessidade com o qual o tempo de execução seria menor.

6. CONCLUSÕES

Primeiramente, a revisão da literatura mostra que existem estudos e mais estudos sobre algoritmos de redução de ruído em sinais de voz, os quais são testados constantemente em diversos artigos científicos em termos de desempenho, complexidade e qualidade de filtragem.

Nesse trabalho foram descritos três algoritmos comumente utilizados, com complexidade entre média e baixa, os quais apresentaram bons resultados nas simulações feitas com o *software* MATLAB.

Em relação a implementação dos algoritmos em DSP, teve-se um bom resultado de filtragem com o algoritmo de subtração espectral, no entanto, houve um desempenho aquém do esperado na velocidade de filtragem devido ao tipo de DSP escolhido e às operações que estes algoritmos utilizam. Para otimização deste tempo de execução, a opção correta seria substituir o DSP atual, que é de ponto fixo, por um de ponto-flutuante, com o qual já haveria melhoras significativas no desempenho.

Por fim, embora neste trabalho não se tenha conseguido bons resultados com a implementação do filtro de Wiener no domínio do tempo, foi visto nas referências que o mesmo é o mais utilizado para filtragem de sinais de voz em tempo real. Isto mostra que o problema está na implementação realizada e não no comportamento do algoritmo em si.

Para concluir, todo o trabalho aqui desenvolvido pode servir de base ou espelho para a construção final de um sistema de controle de ruído em tempo real de sinais de voz, o qual pode ser utilizado nos sistemas de comunicação e diversos outros tipos de aplicações e equipamentos.

7. REFERÊNCIAS

BOLL, Steven. Suppression of acoustic noise in speech using spectral subtraction. **IEEE Transactions on acoustics, speech, and signal processing**, v. 27, n. 2, p. 113-120, 1979.

CHASSAING, Rulph. **DSP applications using C and the TMS320C6x DSK**. 1.ed. John Wiley & Sons, 2003. 360 p.

DHIMAN, Jyoti; AHMAD, Shadab; GULIA, Kuldeep. Comparison between Adaptive filter Algorithms (LMS, NLMS and RLS). **International Journal of Science, Engineering and Technology Research**, v. 2, n. 5, p. 1100, 2013.

EPHRAIM, Yariv; MALAH, David. Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator. **IEEE Transactions on Acoustics, Speech, and Signal Processing**, v. 32, n. 6, p. 1109-1121, 1984.

HU, Yi; LOIZOU, Philipos C. Subjective comparison of speech enhancement algorithms. In: **2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings**. IEEE, 2006. p. I-I.

INGLE, Vinay K; PROAKIS, John G. **Digital Signal Processing Using MATLAB**. 3.ed. Connecticut: Cengage Learning, 2011. 624 p.

ITU - INTERNATIONAL TELECOMMUNICATION UNION. **G.711:** Pulse Code Modulation (PCM) of Voice Frequencies. Geneva, 1972. Fascículo III.4 do Blue Book.

LI, Ze-Nian. DREW, Mark S. **Fundamentals of Multimedia**. 1.ed. Canada: Prentice Hall , 2004. 576 p.

*LIM, Jae S.; OPPENHEIM, Alan V. Enhancement and bandwidth compression of noisy speech. **Proceedings of the IEEE**, v. 67, n. 12, p. 1586-1604, 1979.*

LOIZOU, Philipos C. **Speech Enhancement - Theory and Practice**. 2.ed. Florida: CRC Press, 2013. 711 p.

MATLAB – MATLAB, The Language of Technical Computings. Disponível em: <http://www.mathworks.com/matlabcentral/>. Acesso em: 29 mai. 2016.

MICROCHIP – MPLAB Starter Kit for dsPIC DSCs. Disponível em: <http://www.microchip.com/Developmenttools/>. Acesso em: 21 mai. 2016

OLIVEIRA, André S; ANDRADE, Fernando S. **Sistemas Embarcados – Hardware e Firmware na Prática**. 1.ed. São Paulo: Érica, 2006. 316 p.

OPPENHEIM, Alan V; SCHAFER, Ronald W. **Discrete-Time Signal Processing**. 3.ed. New Jersey: Pearson Prentice-Hall Signal Processing Series, 2009. 1120 p.

ORFANIDIS, Sophocles J. **Optimum signal processing: An introduction.** 2.ed. Collier Macmillan, 1988. 590 p.

SCALART, Pascal. Speech enhancement based on a priori signal to noise estimation. In: **Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on**, 1996. p. 629-632.

SHANNON, Claude Elwood. Communication in the presence of noise. **Proceedings of the IRE**, v. 37, n. 1, p. 10-21, 1949.

SMITH, Julius O. **Spectral Audio Signal Processing**, 1.ed. W3K Publishing, 2011. 674 pages.

UNISAL, Centro Universitário Salesiano de São Paulo. Guia para a elaboração de trabalhos acadêmicos. 4.ed. São Paulo: 2015. 59 p.

UPADHYAY, Navneet; KARMAKAR, Abhijit. Speech Enhancement using Spectral Subtraction-type Algorithms: **A Comparison and Simulation Study. Procedia Computer Science**, v. 54, p. 574-584, 2015.

VASEGHI, Saeed V. **Advanced digital signal processing and noise reduction**. 3.ed. John Wiley & Sons, 2008. 480 p.