UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

ESCOLA POLITÉCNICA

DEPARTAMENTO DE ELETRÔNICA E COMPUTAÇÃO

Remoção de ruído em voz para aplicações em tempo real

Autor: _____
Fabrício Faria Santana

Orientador: _____
Profª. Mariane Rembold Petraglia

Examinador: _____
Prof. José Gabriel Rodrigues Carneiro Gomes

Examinador: _____
Prof. Julio César Boscher Torres

DEL
Dezembro de 2007

“Pouca ciência afasta de Deus,
muita, a Ele reconduz.”

(Louis Pasteur)

A meus pais, fonte de apoio e incentivo na vida universitária.

Agradecimentos

Agradeço a Deus, pela sua providência e misericórdia, que me permitiu chegar ao final deste curso de graduação;

Aos meus pais e seus bons exemplos como pessoas, que contribuíram enormemente para a formação de meu caráter;

À minha esposa Manuella e ao meu filho Gabriel, minha família e meus amores;

Aos grandes mestres com os quais tive a oportunidade de conviver neste período em que me gastei na UFRJ, sem os quais minha vida teria sido menos edificada.

De modo especial, agradeço à minha orientadora neste trabalho, a Professora Mariane, mulher simples e de grande saber científico, pessoa com a qual aprendemos mais do que transformadas e filtros...

Resumo

Este trabalho trata do problema da redução de ruído em voz, de forma a aumentar a sua inteligibilidade. Tem por objetivo estudar alguns processos de filtragem comumente utilizados como técnicas de remoção de ruído em voz para aplicações em telefonia, escutas e videoconferência, por exemplo, de forma a construir algoritmos eficientes para tal função.

A ferramenta de validação das implementações utilizada foi o software MatLab, onde pudemos ter um *background* preciso e rápido para os algoritmos desenvolvidos.

Ao fim de nossos trabalhos, pudemos verificar a eficiência das técnicas descritas na literatura e, comparativamente, pudemos propor um conjunto de técnicas que fosse mais eficiente segundo determinados critérios, como por exemplo, a eficiência computacional e o uso de hardware no meio onde houvesse a necessidade de remoção de ruídos em voz.

Palavras-chave: subtração espectral, filtragem de wiener, *beamformer,* Matlab, processamento de voz.

Conteúdo

Índice de Figuras

1 Introdução

A comunicação é uma necessidade básica do ser humano. Em nossos dias, com os avanços das técnicas de manipulação de semicondutores, observamos grande queda nos preços de aparelhos eletrônicos e conseqüentemente, sua popularização.

Mais do que um ramo ou parte do mercado de microeletrônica, a necessidade de comunicação fez com que o ramo de comunicação, ou muitas vezes denominado ramo de telecomunicações, se tornasse um vasto campo de pesquisa, com aplicações em praticamente todas as áreas da engenharia: construção civil, petróleo, medicina, internet, automobilismo, etc.

A dificuldade de comunicação, normalmente é causada por sinais indesejáveis (ruídos) que acompanham o sinal desejado e que podem ser adicionados no ambiente onde a mensagem foi gerada, ou pelo canal de transporte desta mensagem (ruídos de canal).

Este trabalho se atém ao ramo mais primitivo das telecomunicações que é a necessidade que cada ser humano tem de distinguir palavras na voz de seu semelhante e de se fazer entendido por este. Para isto, serão propostos algoritmos para remoção de ruído em áudio de banda estreita (voz), onde a necessidade é a compreensão da mensagem original. Observaremos que as técnicas utilizadas obrigam-nos a manter um compromisso entre a remoção de ruídos, que dificultam o entendimento da mensagem, e a adição de outros ruídos audíveis porém menos danosos à compreensão da mensagem.

1.1 Motivação

A degradação causada por ruídos incidentes em sinais de voz possui, como produto principal, a diminuição da inteligibilidade dos fonemas e palavras. Inerentemente ao aparelho auditivo humano, possuímos grande capacidade de estimação para atenuarmos estes problemas. Contudo, em situações onde o espectro de freqüências do ruído se aproxima muito do espectro do sinal de voz (ou pode estar em alguma banda capaz de causar algum efeito de mascaramento), nosso aparelho auditivo fica sujeito à perda de confiabilidade pois existem nuances da linguagem onde a entonação e/ou fonemas próximos na freqüência, podem causar distorções na compreensão da mensagem pretendida.

Nas Figuras 1 e 2, observamos exemplos de espectrogramas de sinais de voz. Na Figura 1 observamos o espectrograma do sinal original (a voz de uma mulher pronunciando a frase em inglês *"Thanks for considering me to your upcomming voice project"*) e na Figura 2 observa-se o mesmo áudio corrompido por ruído gaussiano de variância 0,001. O espectrograma disponibiliza a relação existente entre a potência do sinal no tempo e na freqüência.

Nosso objetivo neste trabalho é a criação de um sinal processado com espectrograma semelhante ao do sinal original, ou onde o ruído se desloque para uma região audível porém identificável como sendo ruído, de forma a não dificultar a compreensão da mensagem.

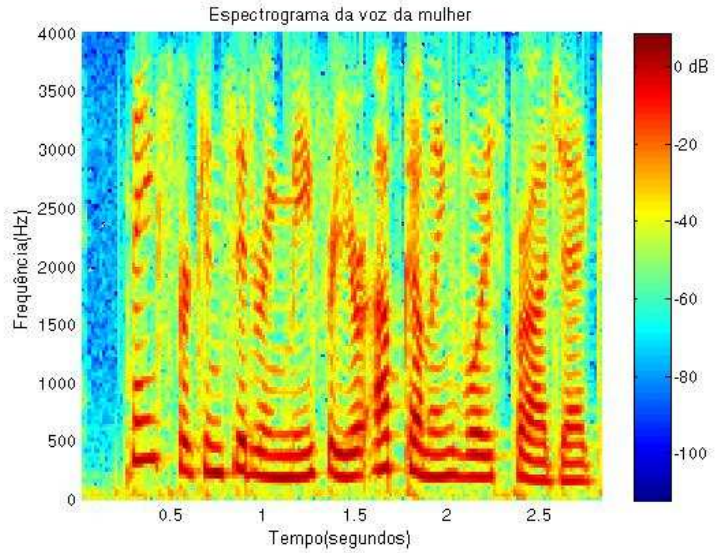


Figura 1: Espectrograma do sinal de voz feminino.

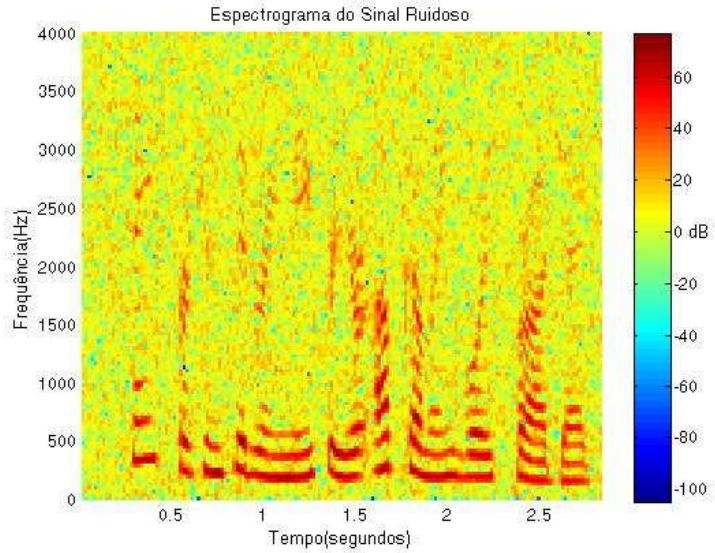


Figura 2: Espectrograma do sinal de voz feminino corrompido.

2 Fundamentos teóricos

Neste capítulo apresentaremos o embasamento teórico no qual se pautou o desenvolvimento dos algoritmos. Faremos um breve comentário sobre filtros, e em seguida serão descritas algumas técnicas de filtragem para remoção de ruído em voz, partindo-se então para o embasamento teórico destas técnicas.

2.1 Filtros

Um filtro, ou estimador, é comumente definido como sendo um sistema projetado para extrair informações sobre uma quantidade de interesse partindo-se de um sinal ruidoso. A partir desta definição geral, podemos particularizar o processo de filtragem deste trabalho, onde fizemos uso somente de filtros digitais.

Basicamente, um sistema de comunicação digital é composto por um transmissor, um canal de transmissão e um receptor, como podemos verificar na Figura 3.

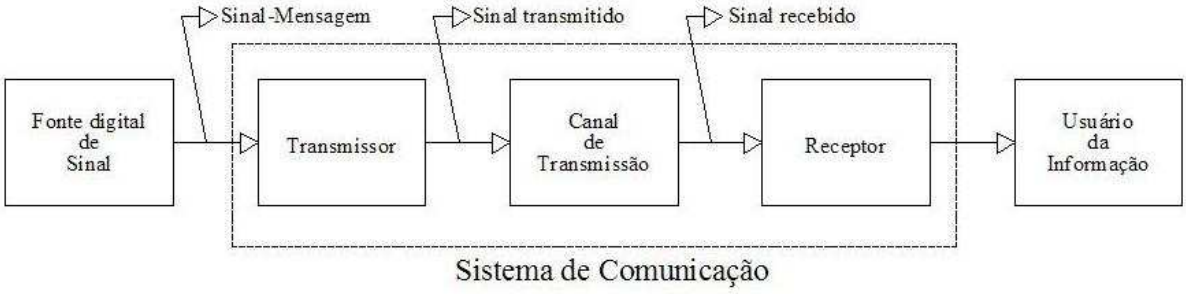


Figura 3: Composição básica de um sistema de comunicação.

O transmissor é o responsável por converter o sinal-mensagem (previamente digitalizado) em formas de onda que possam ser transmitidas pelo canal de transmissão utilizado. O receptor possui a função de receber o sinal transmitido e entregar em sua saída o sinal-mensagem reconstituído.

O processo de filtragem/estimação fica localizado no receptor e sua eficiência será responsável pela realização ou não da comunicação. No caso estudado por nós, podemos expandir o conceito de receptor, pois o receptor final de nosso sistema será o ser humano que intrinsecamente possui a capacidade de "filtragem/estimação" do sinal audível e terá esta capacidade explorada pelos métodos de melhoria de comunicação empregados.

Os filtros digitais são subdivididos em duas categorias, segundo sua implementação, a saber: filtros não recursivos, que possuem resposta ao impulso finita e são chamados filtros FIR (do inglês *Finite Impulse Response*) e filtros recursivos, que em geral possuem resposta infinita ao impulso e são chamados filtros IIR (do inglês *Infinite Impulse Response*).

Na Figura 4, podemos verificar as estruturas de implementação dos filtros FIR (à esquerda) e dos filtros IIR (à direita). Os coeficientes *b* e os *a* definem, respectivamente, os zeros e os pólos dos filtros, onde Z^{-1} representa o atraso unitário.

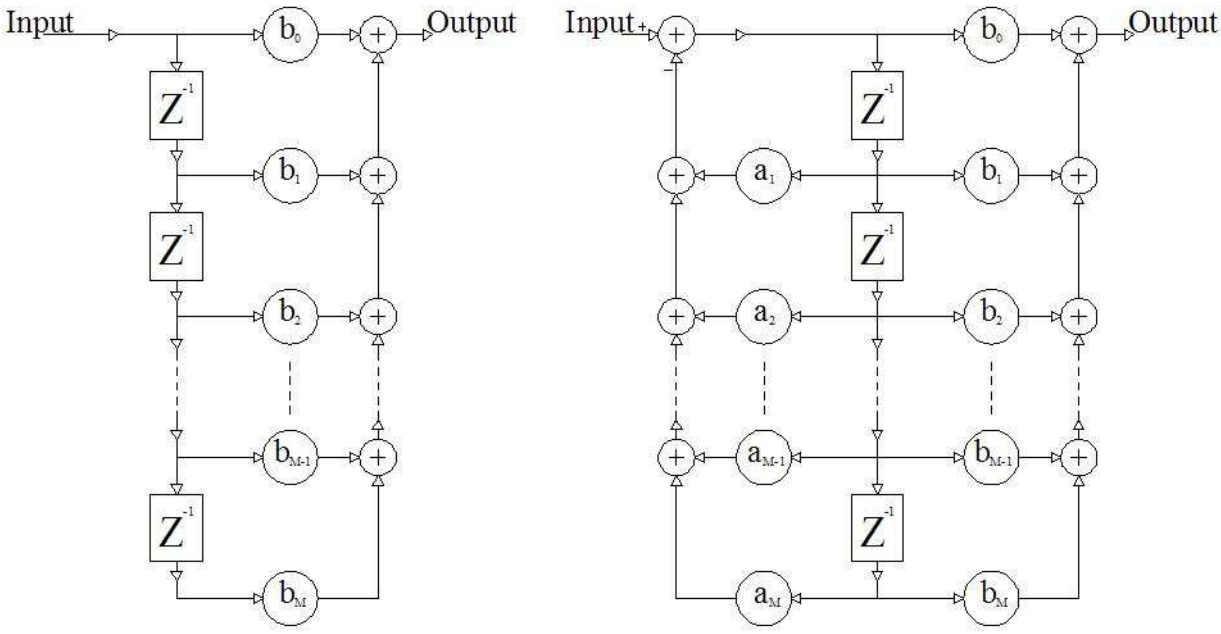


Figura 4: Estrutura dos filtros FIR e IIR.

Em nosso trabalho utilizamos somente filtros FIR devido às técnicas de processamento adotadas, onde se priorizou a facilidade de adaptação dos coeficientes.

2.2 Algumas técnicas de filtragem para redução de ruído e melhorias na qualidade do sinal de voz

A existência de ruído é inevitável em aplicações de processamento de voz em ambientes reais. Num sistema de comunicação, um sinal acústico desejado (como a voz de uma pessoa) captado por um microfone é corrompido por ruído indesejado no ambiente e resulta na formação de um sinal distorcido. Este sinal corrompido necessitará ser processado por um filtro que terá a função de suprimir o ruído, deixando o sinal desejado relativamente inalterado. Este é o conceito básico de cancelamento de ruídos, que rege este trabalho.

O estudo das técnicas de cancelamento de ruído vêm sendo feitos desde 1940 e vários métodos foram propostos e investigados, conforme [1], [2] e [3]. Estes métodos de aproximação podem ser classificados em três categorias básicas: Técnicas de *Beamforming* que exploram o uso de múltiplos sensores, Cancelamento Adaptativo de Ruído que utiliza um sensor primário que capta o sinal ruidoso e um outro sensor utilizado para captar o sinal de referência de ruído, e Modificação Espectral Adaptativa que se aplica a um único sensor.

Beamforming ou formação de feixe: Esta técnica foi desenvolvida para fazer uso de um conjunto de sensores (microfones) que tem a função de explicitar o sinal desejado e suprimir o ruído. Este conjunto de sensores é espacialmente posicionado no ambiente ruidoso onde se quer captar o sinal desejado. A posição e a geometria de espalhamento dos sensores é conhecida e referenciada a um ponto no ambiente. Estes sensores captam o sinal da(s) fonte(s) desejada(s) do ambiente e, através de pesos diferenciados para seus sinais de saída, consegue-se a formação de um feixe direcional de captação de sinal. Desta forma, o sinal que estiver se propagando na direção de interesse será reforçado e os de outras direções, suprimidos.

Cancelamento Adaptativo de Ruído: Está técnica faz uso de dois microfones no ambiente, um para captar o sinal de interesse ruidoso e outro posicionado em local onde a captação do sinal de interesse seja mínima ou nula, tendo este a função de captar o ruído ambiente puro. Desta forma, a técnica tenta recriar adaptativamente, uma réplica do ruído adicionado ao sinal de interesse, partindo-se do ruído de referência captado pelo segundo microfone para posteriormente subtraí-lo do sinal ruidoso do primeiro microfone.

Modificação Espectral Adaptativa: Esta técnica utiliza um único microfone para captação do sinal de interesse e do ruído. De posse do sinal ruidoso, esta técnica tenta restaurar a magnitude espectral do sinal desejado, subtraindo do sinal ruidoso uma estimativa da magnitude espectral do ruído.

2.3 Embasamento teórico das técnicas de remoção de ruído utilizadas neste trabalho

Em nosso trabalho, fizemos uso de duas das três técnicas de remoção de ruído em áudio citadas no item anterior, a saber: *Beamforming* e Modificação Espectral Adaptativa. Na seqüência, explicitaremos de forma mais profunda seus princípios teóricos de atuação.

Na Figura 5 podemos verificar um diagrama esquemático de formação de um feixe direcional, com o uso de um conjunto de microfones uniformemente espaçados posicionados em uma mesma linha imaginária. Deve ser deixado claro que não é esta a única forma de posicionamento geométrico dos microfones, que comumente são dispostos em forma circular, como por exemplo em sonares de submarinos.

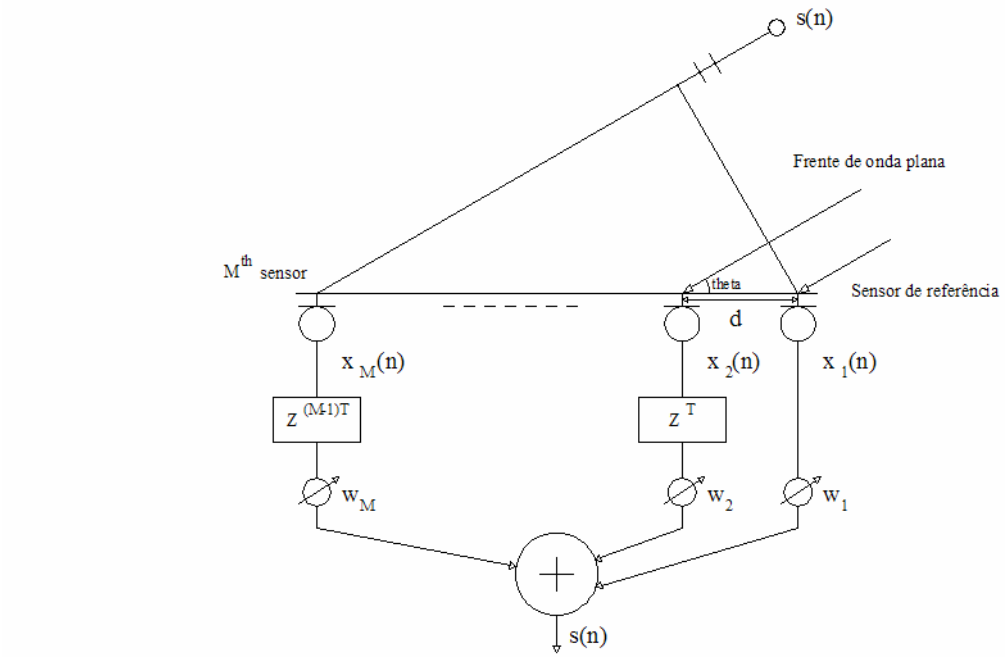


Figura 5: Ilustração de um *beamformer* com microfones uniformemente espaçados.

Na Figura 5 o sistema considerado possui *M* microfones, $x_m(n)$ é o sinal recebido no m-ésimo sensor, w_m é um peso atribuído a cada sensor, *T* é o atraso relativo entre microfones adjacentes, *theta* é o ângulo de incidência da frente de onda na linha onde estão posicionados os microfones, e $v_m(n)$ é o ruído captado por cada microfone, que é assumido como sendo não correlacionado com o sinal desejado *s(n).*

Este *beamformer* costuma ser tipificado pela sua característica de processamento do tipo atraso e soma. Seu funcionamento é baseado num estudo primário da geometria do problema, onde podemos verificar que *d*, a distância constante entre os microfones, está relacionada com *T* e o ângulo *theta* pela relação abaixo, onde *c* é a velocidade de propagação.

$$T = d\cos(theta)/c \tag{1}$$

Um ponto muito importante a ser considerado é a modelagem de onda plana atribuída ao sinal sonoro. Para tal, devemos considerar que a distância da fonte sonora ao conjunto de microfones é suficientemente grande, condição esta facilmente atendida já que o arranjo de microfones é montado no ambiente onde se localiza a fonte sonora e não em um acessório de mão por exemplo.

Podemos modelar a saída dos sensores da seguinte forma:

$$x(n) = \begin{bmatrix} x_1(n) \\ x_2(n) \\ \dots \\ x_M(n) \end{bmatrix} = \begin{bmatrix} s(n) \\ s(n-T) \\ \dots \\ s(n-(M-1)T) \end{bmatrix} + \begin{bmatrix} v_1(n) \\ v_2(n) \\ \dots \\ v_M(n) \end{bmatrix} \tag{2}$$

Aplicando a transformada de Fourier, teremos:

$$\begin{bmatrix} X_1(jw) \\ X_2(jw) \\ \dots \\ X_M(jw) \end{bmatrix} = \begin{bmatrix} S(jw) \\ S(jw)e^{-jwT} \\ \dots \\ S(jw)e^{-jw(M-1)T} \end{bmatrix} + \begin{bmatrix} V_1(jw) \\ V_2(jw) \\ \dots \\ V_M(jw) \end{bmatrix} \tag{3}$$

A potência do sinal recebida no m-ésimo microfone é dada por:

$$P_m(w) = E\left[\left|X_m(jw)\right|^2\right] = E\left[\left|S(jw)\right|^2\right] + E\left[\left|V_m(jw)\right|^2\right] \tag{4}$$

onde *E[.]* denota a expectativa matemática. Assume-se então que todos os microfones possuem o mesmo espectro de potência de ruído, ou seja:

$$P_1(w) = P_2(w) = \dots = P_M(w) \tag{5}$$

O *beamformer* de atrasos e somas consiste então em se aplicar um atraso T_m e um peso na amplitude w_m à saída do m-ésimo sensor. Em seguida, somam-se os M sinais resultantes. A expressão geral de um *beamformer* de atrasos e somas fica então com a forma:

$$Y(jw) = \sum_{m=1}^{M} w_m X_m(jw) e^{jwTm} \tag{6}$$

onde w_m é um vetor com os pesos dos microfones.

Pela substituição de (3) em (6), e forçando-se $w_m = 1/M$ e $T_m=(m-1)T=(m-1).d.cos(theta)/c$ teremos:

$$Y(jw) = S(jw) + \frac{1}{M}\sum_{m=1}^{M} V_m(jw) e^{jw(m-1)T} \tag{7}$$

que comprova a eficiência do processamento na redução do ruído no sinal processado $y(n)$, que será proporcional ao número de sensores utilizados.

Para a Modificação Espectral, fizemos uso de duas técnicas descritas na literatura, que são a Subtração Espectral e o Filtro de Wiener Paramétrico. Ambas as técnicas podem ser esquematizadas pela Figura 6.

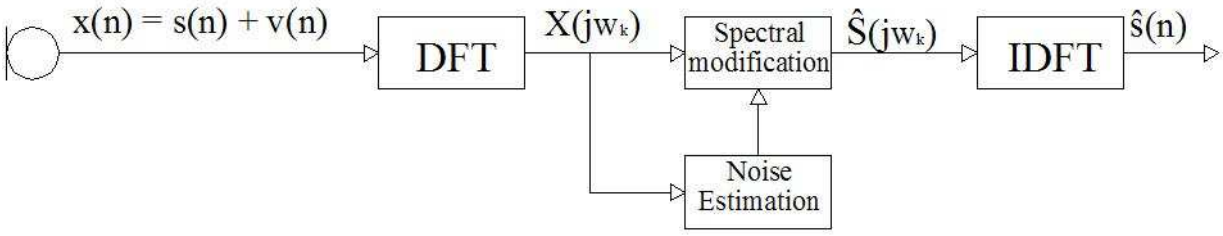


Figura 6: Realçando a voz com a modificação espectral.

Pelo modelo de processamento da Figura 6, podemos escrever:

$$X(jw_k) = S(jw_k) + V(jw_k) \tag{8}$$

onde S é a transformada de Fourier do sinal desejado, V é a transformada do ruído e X é a transformada do sinal corrompido. Conseqüentemente podemos escrever:

$$|X(jw_k)|^2 = |S(jw_k)|^2 + |V(jw_k)|^2 + 2|S(jw_k)||V(jw_k)|\cos\theta_k \tag{9}$$

$$|X(jw_k)|^2 = S^2(w_k) + V^2(w_k) + 2S(w_k)V(w_k)\cos\theta_k \tag{10}$$

onde $S(w_k)$ e $V(w_k)$ representam as magnitudes do sinal desejado e do ruído, respectivamente, e θ_k é a diferença de fase entre o sinal de voz e o ruído.

Se o sinal de voz e o ruído são processos randômicos, estacionários e não correlacionados, (10) pode ser aproximada por:

$$X^2(w_k) = S^2(w_k) + V^2(w_k) \tag{11}$$

Sobre tais circunstâncias, o espectro de potência instantâneo (ou a magnitude quadrada do sinal), $S^2(w_k)$, pode ser recuperado pela subtração de uma estimativa da magnitude quadrada do ruído.

$$\begin{aligned} \hat{S}^2(w_k) &= X^2(w_k) - \hat{V}^2(w_k) \\ &= S^2(w_k) + \left[V^2(w_k) - \hat{V}^2(w_k) \right] \end{aligned} \tag{12}$$

Entretanto, a magnitude do espectro do sinal de voz é obtida por:

$$\begin{aligned} \hat{S}(w_k) &= \sqrt{\hat{S}^2(w_k)} \\ \hat{S}(w_k) &= \sqrt{X^2(w_k) - \hat{V}^2(w_k)} \end{aligned} \tag{13}$$

Neste ponto, a técnica de subtração espectral faz uso da propriedade que o aparelho auditivo humano possui de ser pouco crítico a variações pequenas de fase. Desta forma, combinamos (13) com a fase do sinal ruidoso, sem nos preocuparmos em estimar a fase do ruído.

$$\hat{S}(w_k) = \left(\sqrt{X^2(w_k) - \hat{V}^2(w_k)} \right) e^{j\psi k} \tag{14}$$

Esta é a forma básica do método de redução de ruído popularmente chamado de Subtração Espectral. Um algoritmo similar pode ser desenvolvido somente no domínio da magnitude espectral. Se o ruído e o sinal desejado não são correlacionados, como propomos anteriormente, podemos então escrever:

$$X(w_k) = S(w_k) + V(w_k) \tag{15}$$

Conseqüentemente, a magnitude espectral do sinal de voz pode ser diretamente estimada por:

$$\hat{S}(w_k) = X(w_k) - \hat{V}(w_k)$$
$$\hat{S}(w_k) = S(w_k) + \left[V(w_k) - \hat{V}(w_k) \right] \quad (16)$$

Numa forma mais geral, (12) e (16) podem ser expressas por:

$$\hat{S}^b(w_k) = X^b(w_k) - \eta \hat{V}^b(w_k) \quad (17)$$

onde *b* é um expoente inteiro, que usamos igual a *2* para a estimativa da magnitude quadrada, e η é um parâmetro para controle da quantidade de ruído a ser retirada na subtração espectral, onde $\eta=1$ se traduz na retirada máxima de ruído. Conseqüentemente, a estimativa do espectro da voz será construída por:

$$\hat{S}(jw_k) = \left[X^b(w_k) - \eta \hat{V}^b(w_k) \right]^{\frac{1}{b}} e^{j\varphi k} \quad (18)$$

A expressão (18) é conhecida como a subtração espectral paramétrica, onde para *b=2* e $\eta=1$ temos a chamada *IPS* (do inglês *Instantaneous Power spectral Subtraction)* e para *b=1* e $\eta=1$ teremos a chamada *MS* (do inglês *Magnitude spectral Subtraction*). Trabalhamos sempre com a *IPS* em nossas implementações de subtração espectral.

Para a estimativa da magnitude do ruído, se fez necessário implementar um sistema de decisão capaz de estimar o ruído em regiões do sinal ruidoso onde não houvesse presença de voz. Fizemos uso então de um sistema, que segmenta o sinal ruidoso em blocos de *N* amostras, transforma estes blocos via *DFT* em blocos com *N* amostras espectrais e, utilizando uma média recursiva, estima o ruído nos blocos através de considerações sobre o ruído no bloco anterior, ou seja:

$$\hat{V}_t^b(w_k) = \begin{cases} \alpha_a \hat{V}_{t-1}^b(w_k) + (1-\alpha_a) X_t^b(w_k) \because X_t^b(w_k) \geq \hat{V}_{t-1}^b(w_k) \\ \alpha_d \hat{V}_{t-1}^b(w_k) + (1-\alpha_d) X_t^b(w_k) \because X_t^b(w_k) < \hat{V}_{t-1}^b(w_k) \end{cases} \quad (19)$$

Nesta expressão, *b* é igual a *2*, para termos a magnitude quadrada. Os parâmetros α_a e α_d representam respectivamente os coeficientes *attack* e *decay*. O coeficiente de ataque α_a deve ser bem pequeno para evitar que atenuemos o início de palavras, onde existe uma transição da condição de ausência de voz para presença de voz. O coeficiente de decaimento α_d deve ser um pouco maior, pois geralmente a envoltória do sinal de voz na transição de períodos de presença de voz para ausência de voz é mais lento, ou seja, a envoltória do sinal de voz no início de palavras é mais brusco que ao final das mesmas.

Sendo assim, os valores dos alfas têm relação direta com o tamanho dos blocos e com a freqüência de amostragem, como pode ser constatado nos códigos-fonte do Apêndice.

Também é preciso mencionar que foi realizado um janelamento (com *overlap* de 50%) nos blocos do sinal, de forma a suavizarmos a sua reconstrução. A janela utilizada foi a janela de Hanning, sendo que outras janelas, como a de Hamming também obtiveram bons resultados.

A diferença básica entre a Subtração Espectral e o Filtro Paramétrico de Wiener é a que o filtro de Wiener é concebido para minimizar o erro médio quadrático. A solução geral para o Filtro de Wiener não-causal tem a forma:

$$H_o(w) = \frac{S_{sy}(w)}{S_x(w)} \tag{20}$$

onde o S aqui significa espectro de potência.

Se *s(n)* e *v(n)* não são correlacionados, teremos:

$$H_o(w) = \frac{S_s(w)}{S_x(w)} \tag{21}$$

Podemos reescrever (21) com a forma:

$$H_o(W) = \frac{\left(S_x(w) - S_v(w)\right)}{S_x(w)} = \frac{1}{1 + \frac{1}{SNR(w)}} \tag{22}$$

que nos mostra a característica de atuação mais forte em regiões de baixa relação sinal-ruído que o Filtro de Wiener possui. Em regiões de alta relação sinal-ruído, observamos que o filtro praticamente não altera o sinal de entrada.

De posse deste conhecimento e de forma semelhante à subtração espectral, podemos definir um operador para estimar o sinal desejado partindo-se do sinal ruidoso, que chamaremos então de filtro de Wiener paramétrico.

$$\hat{S}(jw_k) = \hat{H}_{PW}(w_k)X(jw_k) \tag{23}$$

sendo:

$$\hat{H}_{PW}(w_k) = \left[\frac{X^b(w_k) - \eta \hat{V}^b(w_k)}{X^b(w_k)}\right]^{\frac{1}{b}} \tag{24}$$

onde η é um parâmetro que definirá a quantidade de ruído a ser reduzido, que varia de zero a um, sendo que utilizamos $\eta=1$ que é o máximo.

Assim como a estimação do ruído é o coração da subtração espectral, a computação do ganho do filtro para cada freqüência é o problema central do filtro paramétrico de Wiener. Vamos definir então a relação sinal ruído *a posteriori* como sendo:

$$E(w_k) = \frac{X(w_k)}{V(w_k)} \tag{25}$$

Estão, reescrevemos (24) com a forma:

$$\hat{H}_{PW}(w_k) = \left[1 - \eta \frac{1}{E^b(w_k)}\right]^{\frac{1}{b}} \tag{26}$$

Discutimos anteriormente, na subtração espectral, um método para estimar o espectro do ruído. Entretanto, no filtro de Wiener paramétrico temos a necessidade de estimar a relação sinal ruído *a posteriori*, $E(w_k)$.

Todavia, a estimativa da relação sinal ruído flutua de acordo com a variância do espectro da *DFT*, conforme [6]. Duas aproximações podem ser empregadas para reduzir esta flutuação: fazendo-se uma média temporal e/ou uma média na freqüência no espectro da DFT antes de se computar a relação sinal ruído. No nosso trabalho fizemos uso somente da média temporal do espectro da DFT, pois nosso áudio é de banda estreita e a média na freqüência não trouxe grandes benefícios.

Se nomearmos $X_t(w_k)$ a magnitude do espectro do sinal ruidoso no quadro de índice *t* (supondo uma divisão em quadros de tamanho curto do sinal original no tempo), a média temporal pode ser implementada de forma parecida com a estimação do ruído, onde usamos uma recursão de um pólo:

$$\bar{X}_t(w_k) = \begin{cases} \beta_a \bar{X}_{t-1}(w_k) + (1-\beta_a) X_t(w_k) \because X_t(w_k) \geq \bar{X}_{t-1}(w_k) \\ \beta_d \bar{X}_{t-1}(w_k) + (1-\beta_d) X_t(w_k) \because X_t(w_k) < \bar{X}_{t-1}(w_k) \end{cases} \quad (27)$$

onde β_a e β_d possuem a mesma função que os parâmetros α_a e α_d da estimação do ruído porém possuem valores diferentes. Com a combinação de (19) e (27) poderemos então estimar a relação sinal ruído para sinais de banda estreita (no inglês *narrow-band*) no instante *t* como:

$$E_t^N(w_k) = \frac{\bar{X}_t(w_k)}{\hat{V}_t(w_k)} \quad (28)$$

É importante salientar, que as técnicas de remoção de ruído que realizam modificação espectral, introduzem um ruído denominado ruído musical que tende a ser menos danoso à compreensão da mensagem de voz do que os ruídos de ambiente mais comuns, como constatamos em nossas implementações.

De posse desta base teórica, implementamos algoritmos no Matlab que realizassem o processamento descrito pelas técnicas.

3 Implementação dos algoritmos

A meta de nossas implementações se deteve em criarmos uma mescla entre as técnicas de remoção de ruído com modificação espectral e o uso de um *beamformer*. Todos os algoritmos estarão impressos ao final deste trabalho no apêndice.

Para todos testes, utilizamos sempre arquivos de áudio com as seguintes características: 128 Kbps de taxa de bits, 16 bits de tamanho da amostra, 1 canal (mono), 8 Khz de taxa de amostragem e com formato *PCM* (*Pulse Code Modulation*). Estão disponíveis no Departamento de Eletrônica da UFRJ (DEL) os áudios utilizados, bem como os resultados.

Nossas implementações foram realizadas em ambiente *offline*, mas todas foram pensadas e idealizadas para ambientes e aplicações em tempo-real. As simulações no *Matlab* levaram este fato em consideração.

Iniciamos nosso trabalho pela implementação da subtração espectral. Basicamente, nossa implementação possui um dispositivo intrínseco que realiza buscas por intervalos do sinal ruidoso onde não se observa a presença de voz, comumente conhecido na literatura pela sigla VAD (do inglês *Voice Activity Detect*). Desta forma, podemos estimar o ruído eficientemente, pois entre palavras, e até mesmo entre sílabas, a voz humana possui menor potência e o ruído aditivo se sobressai.

Como podemos observar em (19), a estimativa do ruído, feita pelo VAD, acontece no quadro atual para ser aplicada no quadro seguinte. Esta estimativa baseia-se no fato de que o ruído varia pouco entre quadros adjacentes e sua eficiência aumenta com a estacionariedade do ruído.

Em um segundo momento, realizamos a implementação do filtro de Wiener paramétrico, conforme a descrição teórica do capítulo anterior. Observa-se um maior custo computacional do filtro de Wiener paramétrico, devido ao cálculo realizado para satisfazermos (27). Observaremos nas simulações que este custo computacional não oferece resultados muitos superiores aos da subtração espectral.

Finalmente, implementamos a remoção de ruído com uso do *array* de microfones, onde pudemos constatar que a eficiência da técnica está amarrada ao número de microfones componentes do array.

Como previsto no início dos trabalhos, optamos pela realização de testes de forma a mesclar as técnicas estudadas para um melhor resultado na remoção de ruídos.

Abandonamos os estudos com o Wiener paramétrico e escolhemos a Subtração Espectral para ser a nossa técnica de modificação espectral padrão. Como já foi mencionado, o filtro de Wiener paramétrico não oferece ganho significativo na qualidade do áudio, para a duplicação do custo computacional em relação à Subtração Espectral que ele necessita.

Inicialmente ponderamos se era viável a aplicação de uma Subtração Espectral em cada um dos microfones do *array*. Implementamos desta forma, mas observamos que o ruído musical adicionado a cada microfone pela técnica da Subtração Espectral, causava um efeito desconfortável no sinal processado. Como não queríamos aumentar o custo computacional, esta hipótese foi abandonada, pois aumentaríamos o custo computacional da Subtração Espectral linearmente com o incremento do número de microfones do *array*.

A mescla de técnicas que concluímos ser mais eficiente é composta de um *array* de microfones seguido de uma subtração espectral. Mostraremos, que o ganho de eficiência com este conjunto, reduz a necessidade de uso de microfones no *array*, apesar de proporcionar melhores resultados na remoção de ruídos.

4 Resultados Experimentais

A eficiência de nosso algoritmo para realização da subtração espectral pode ser comprovada pelas Figuras seguintes. Na Figura 7, observamos a entrada de nosso sistema em azul, onde a voz de uma mulher foi adicionada de ruído gaussiano com variância 0,001, acompanhada da saída processada em vermelho.

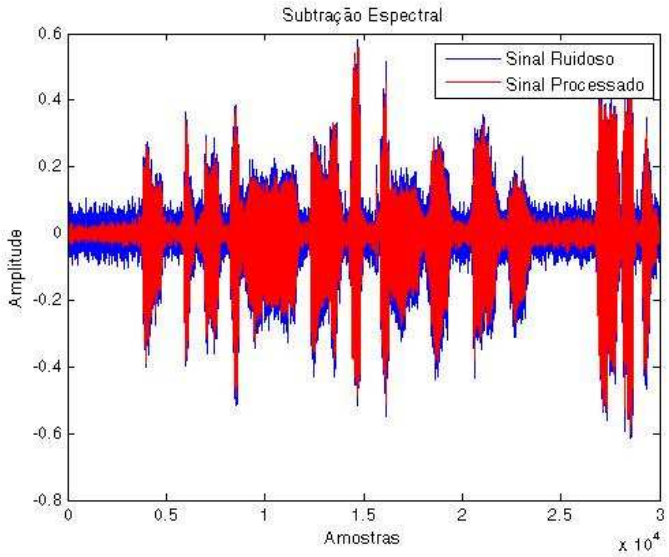


Figura 7: Removendo ruído com a modificação espectral.

Na Figura 8, os espectrogramas dos sinais ruidoso e processado, respectivamente.

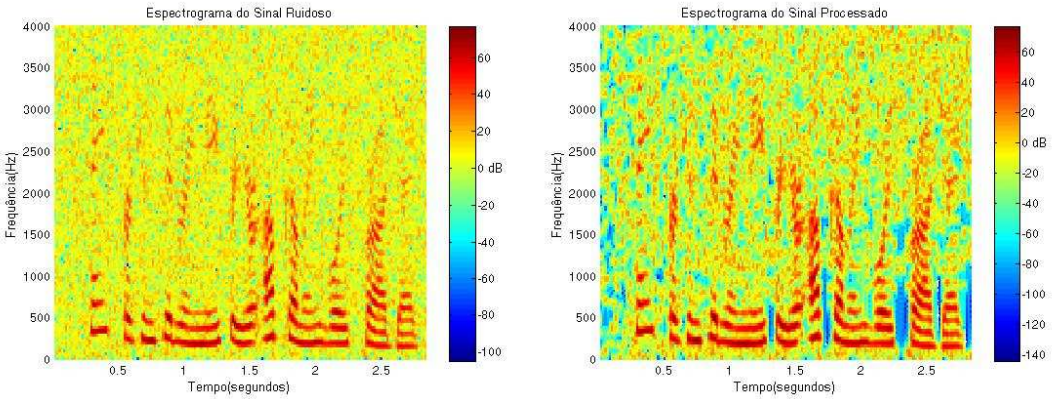


Figura 8: Espectrogramas do sinal ruidoso e processado – subtração espectral.

Para melhor medida qualitativa, observamos na Figura 9 a comparação dos espectrogramas do sinal original sem ruído e do sinal processado, respectivamente. Este sinal de voz é o mesmo citado no início deste texto: uma voz feminina falando a frase em inglês *"Thanks for considering me to your upcomming voice project"* .

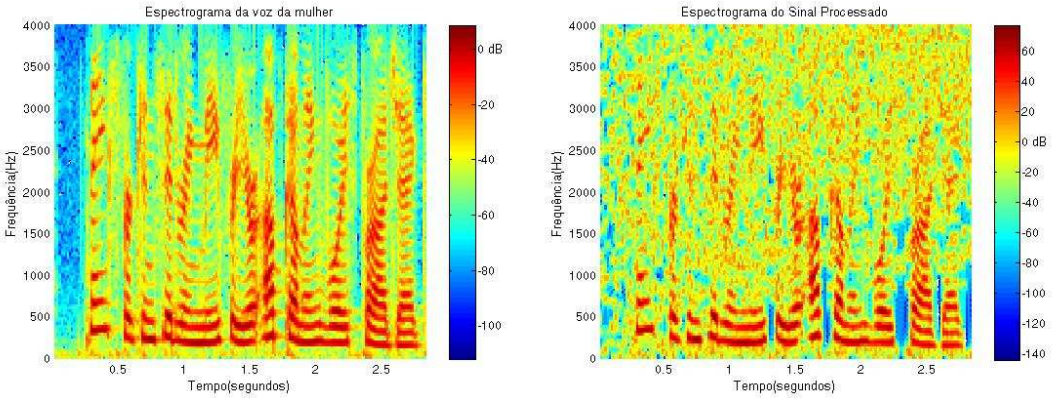


Figura 9: Espectrogramas do sinal original limpo e processado – sub. espectral.

Apresentaremos agora na Figura 10, a eficiência na remoção de ruído nas mesmas situações para o filtro de Wiener paramétrico, onde em azul está o sinal de entrada e em vermelho o sinal processado.

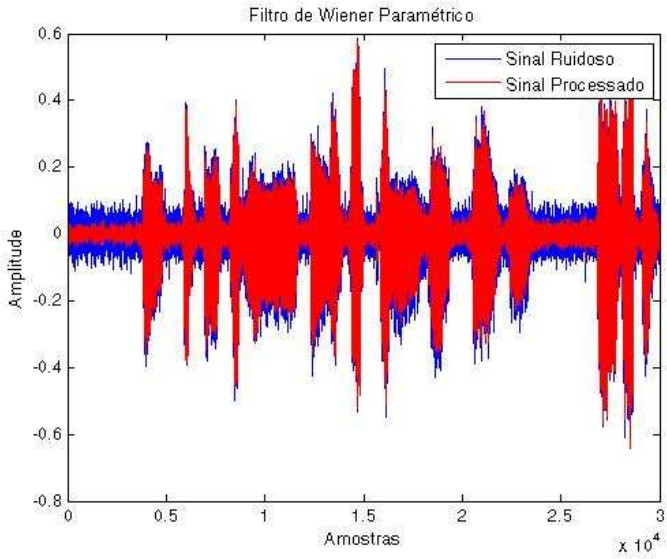


Figura 10: Removendo ruído com o filtro de Wiener paramétrico.

Segue-se agora na Figura 11 os espectrogramas do sinal de entrada ruidoso e o de saída do filtro de Wiener, à direita.

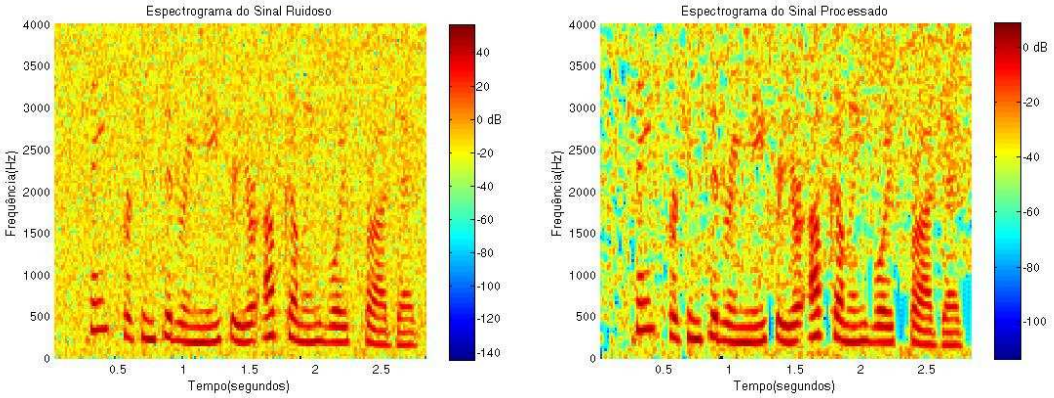


Figura 11: Espectrogramas do sinal ruidoso e processado – filtro de Wiener.

Novamente comparamos a performance com o sinal limpo original agora no Wiener.

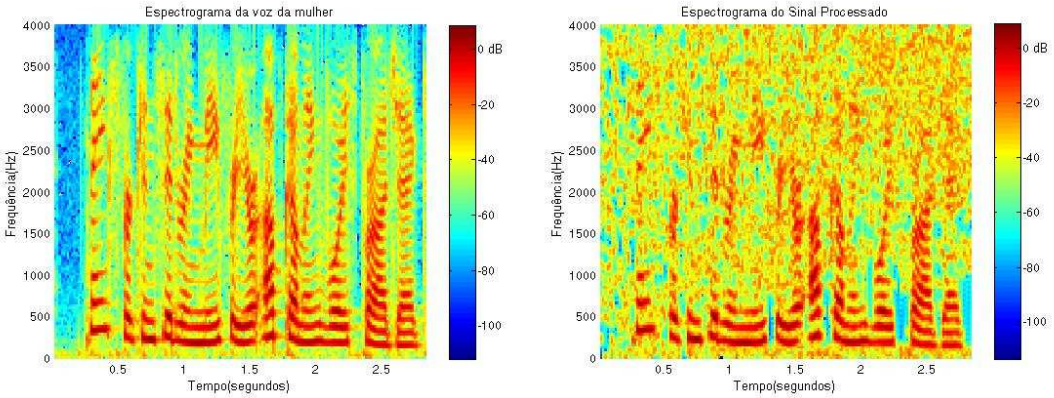


Figura 12: Espectrogramas do sinal original limpo e processado – filtro de Wiener.

Podemos constatar que é bastante difícil mensurar a maior eficiência do filtro de Wiener, que pela teoria existe.

Agora passamos para os resultados da implementação do *array* de microfones. Nestas implementações usamos um *array* com 10 microfones.

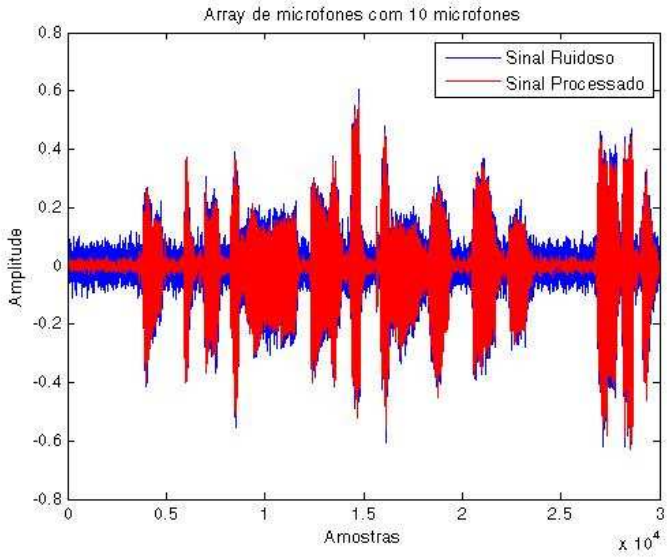


Figura 13: Removendo ruído com *array* de 10 microfones.

Na Figura 14, observamos os espectrogramas do sinal de entrada e do sinal processado pelo *array* com 10 microfones.

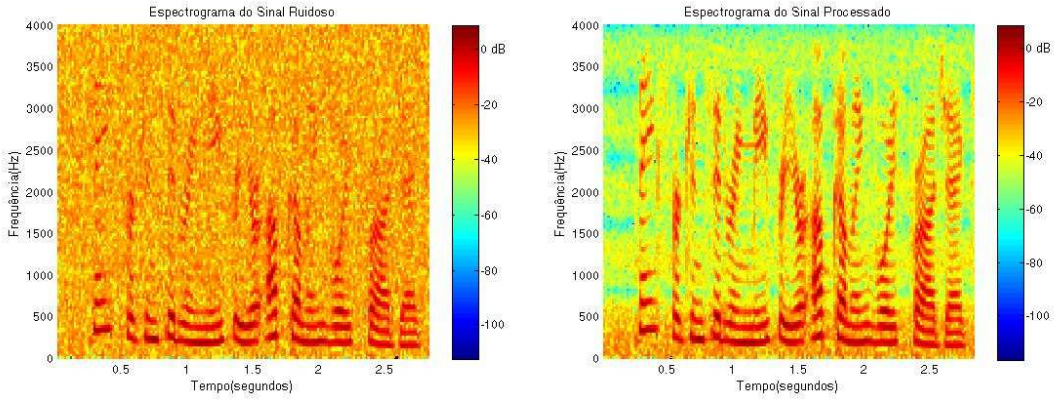


Figura 14: Espectrogramas do sinal ruidoso e processado – *array* de 10 microfones.

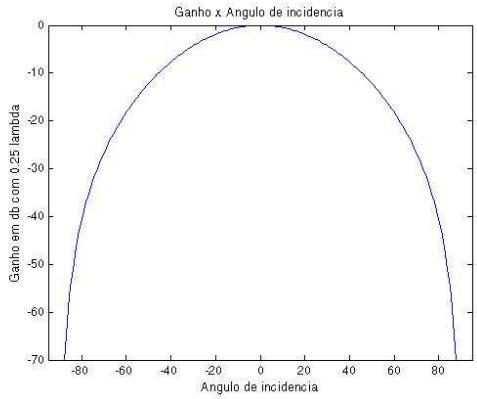
Em se tratando do *array*, devemos fazer mais algumas ponderações. Não é a intenção deste trabalho a identificação de fontes, mas o uso de um *array* de microfones possui aplicações nesta área. Como sua característica é a de reforçar o sinal proveniente de determinada direção, os sinais provenientes de outras direções são atenuados. Desta forma, em uma ambiente com diversas pessoas falando, é possível o direcionamento do feixe auditivo do *array* a fim de reforçar ou identificar a voz e as palavras de determinada pessoa.

Também devemos ponderar sobre o espaçamento máximo entre os microfones do *array* para que não ocorra *aliasing* na captação do sinal. Este espaçamento é dado pela condição [5]:

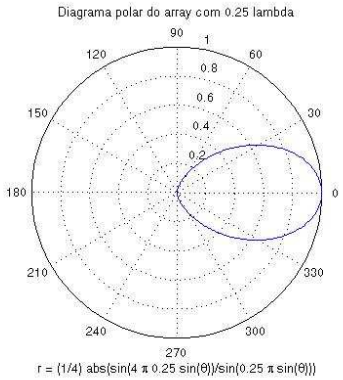
$$d \leq \frac{\lambda_o}{2} \tag{29}$$

No nosso caso, escolhemos d=0,05 m, para que pudéssemos captar freqüências de até 3400 Hz (sendo 340 m/s a velocidade do som) sem *aliasing*. Freqüências acima de 3400 Hz até 4000 Hz correspondem à banda de transição do corte superior dos filtros de telefonia.

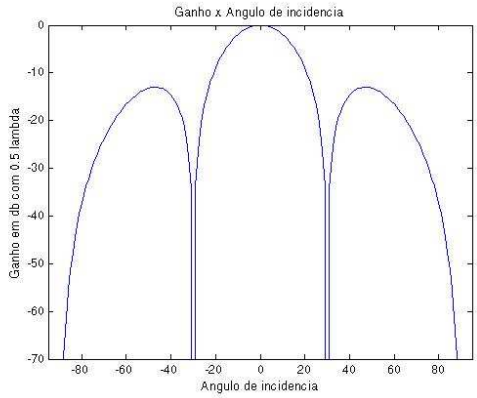
Na Figura 15, podemos fazer uma rápida observação da relação que o espaçamento dos microfones possui na diretividade e no ganho do *array*.



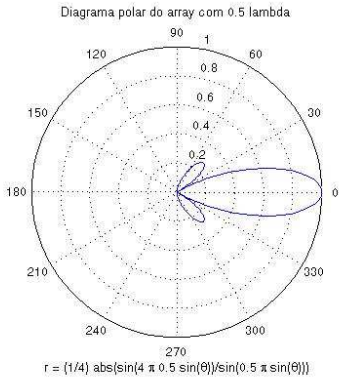
(a)



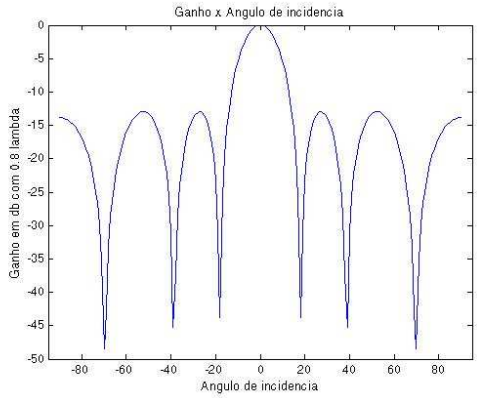
(b)



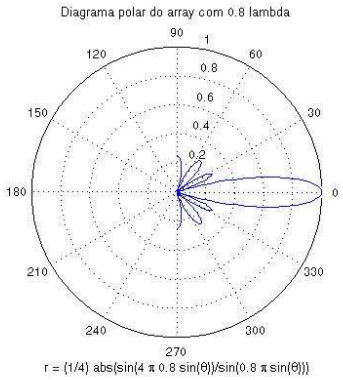
(c)



(d)



(e)



(f)

Figura 15: Tipos de ganho (planar e polar) de um *array* de 4 microfones: (a) planar com *d=0,25 λ* , (b) polar com *d=0,25 λ ,* (c) planar com *d=0,5 λ* , (d) polar com *d=0,5 λ,* (e) planar com *d=0,8 λ* , (f) polar com *d=0,8 λ* .

Após este breve estudo sobre *arrays* de microfones, segue-se um exemplo de uso do *array* na identificação de fontes. O modelo foi o de um automóvel "Fusca", onde havia uma mulher no carona e um motorista homem. O *array* com 10 microfones foi modelado no pára-sol do carona.

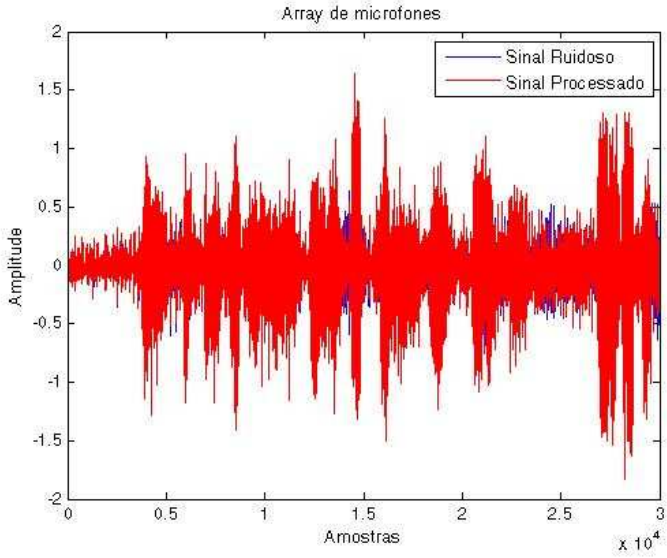


Figura 16: Identificando fontes com *array* de 10 microfones.

Observamos que esta Figura não nos diz muito a respeito da identificação de fontes. Partimos então para a análise dos resultados da Figura 17, que contém os espectrogramas.

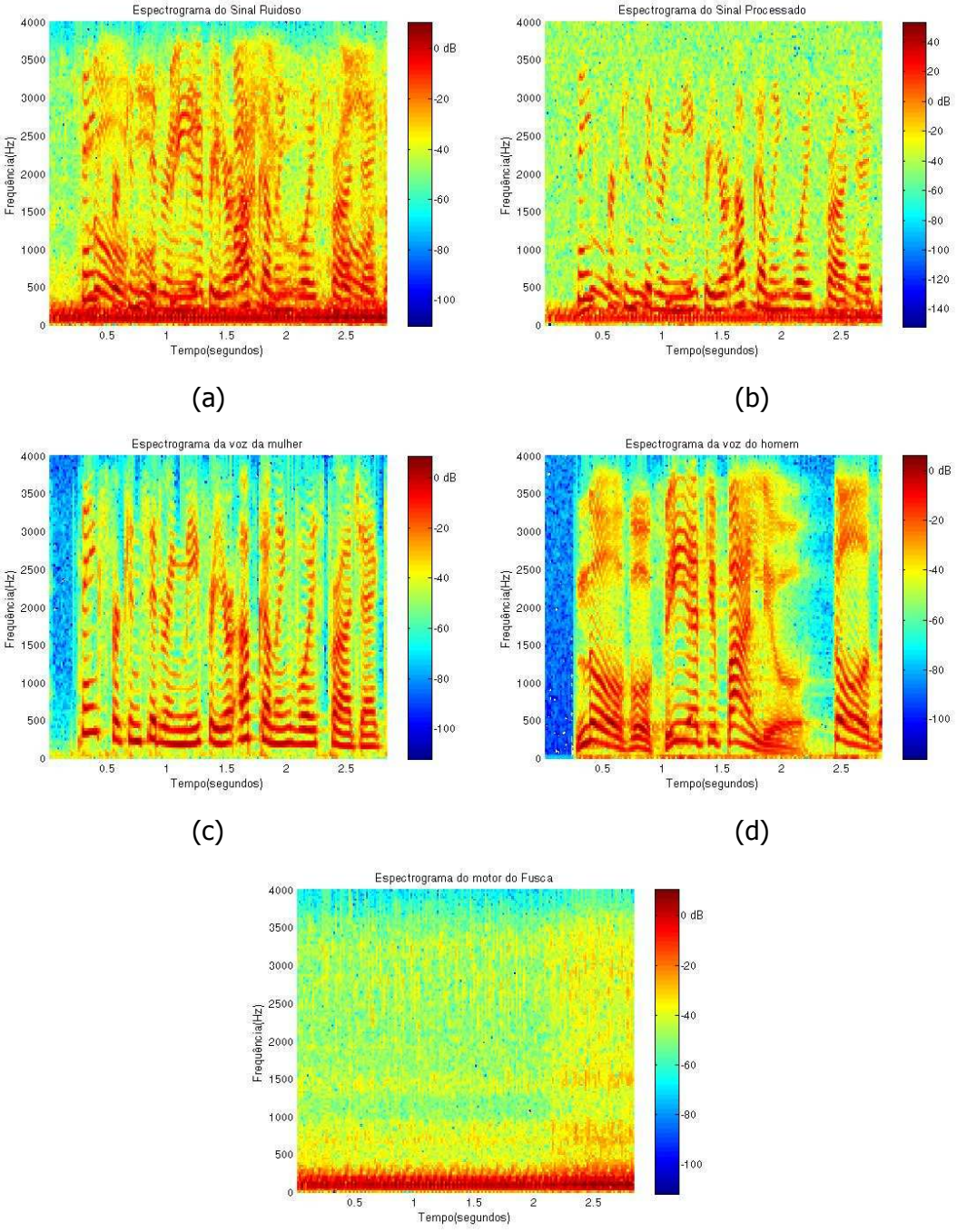


Figura 17: Identificação de fontes com um array de 10 microfones: (a) Sinal ruidoso com homem+mulher+fusca; (b) Sinal processado, priorizando a mulher; (c) Original voz da mulher, (d) Original voz do homem; (e) Motor do Fusca.

Finalmente, passamos ao processamento do sinal ruidoso com o uso de um *array* de 4 microfones em conjunto com uma subtração espectral em sua saída. Verifiquemos a Figura 18.

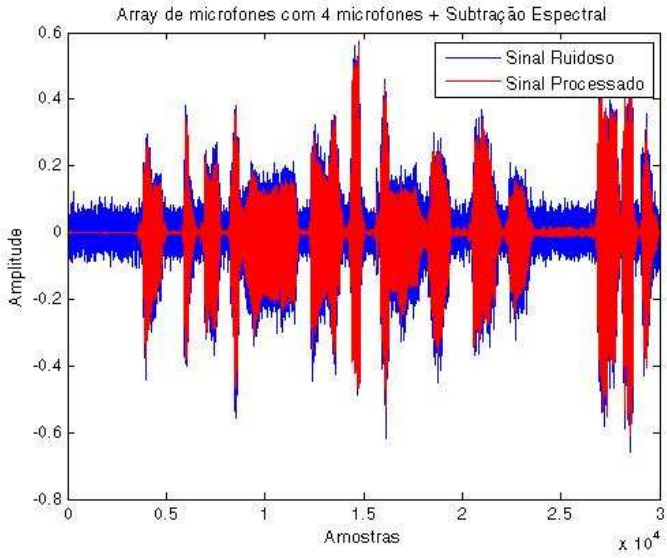


Figura 18: Removendo ruído com *array* de 4 microfones + subtração espectral.

Visualizamos agora na Figura 19 a performance deste sistema pelos espectrogramas dos sinais.

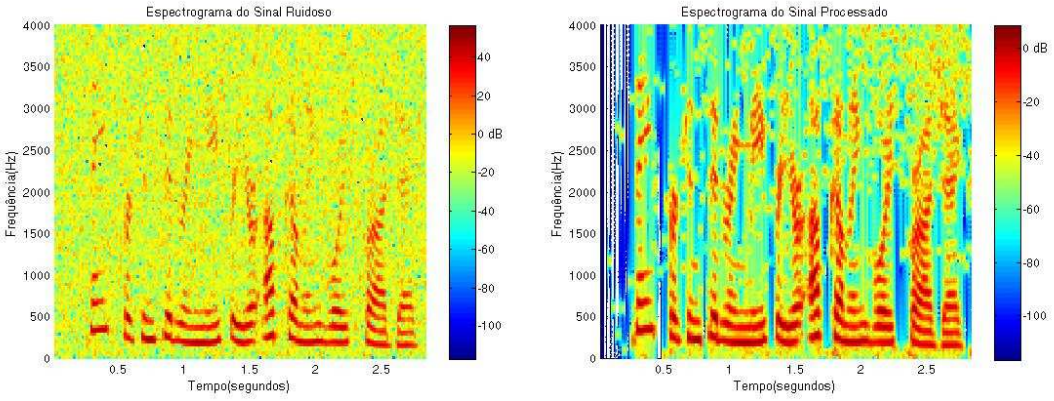


Figura 19: Espectrogramas do sinal ruidoso e processado – *array* + subtração espectral.

Comparamos o sinal resultante com o sinal original limpo na Figura 20.

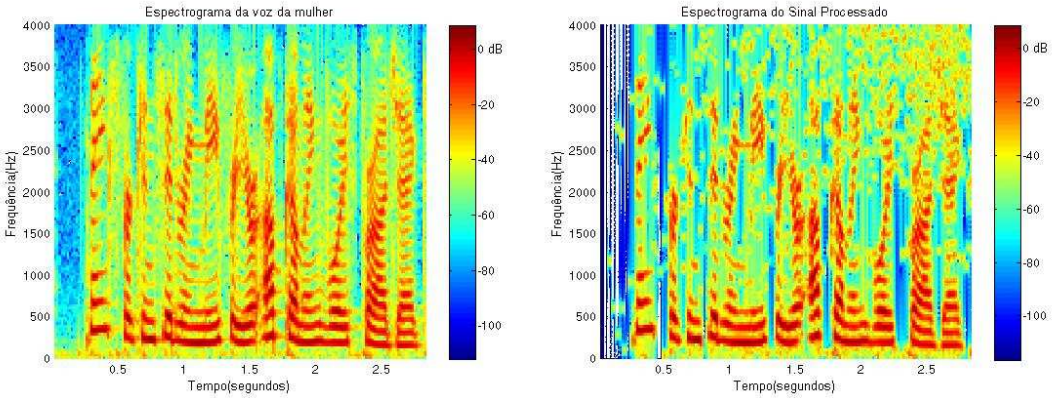


Figura 20: Espectrogramas do sinal limpo e processado – *array* + subtração espectral.

Comparamos também, na Figura 21, os sinais processados pelo *array* de 10 microfones e o deste conjunto de técnicas: *array* de 4 microfones + subtração espectral em sua saída.

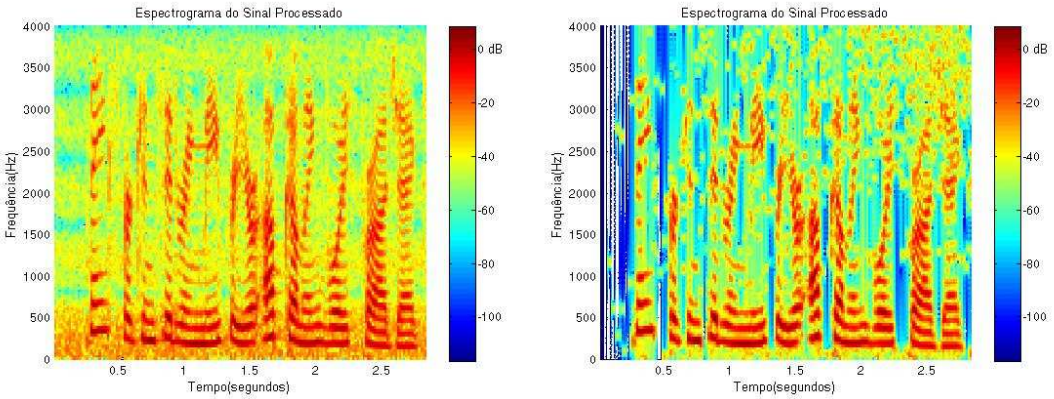


Figura 21: Espectrogramas dos sinais processados pelo *array* 10 microfones e do *array* 4 microfones + subtração espectral.

Para exemplificarmos o uso desta técnica em um mercado de escutas ou espionagem, por exemplo, modelamos um saguão de aeroporto onde se desejou identificar uma voz de mulher. Observamos na Figura 22 os espectrogramas referentes a este modelo, que se assemelha ao do Fusca.

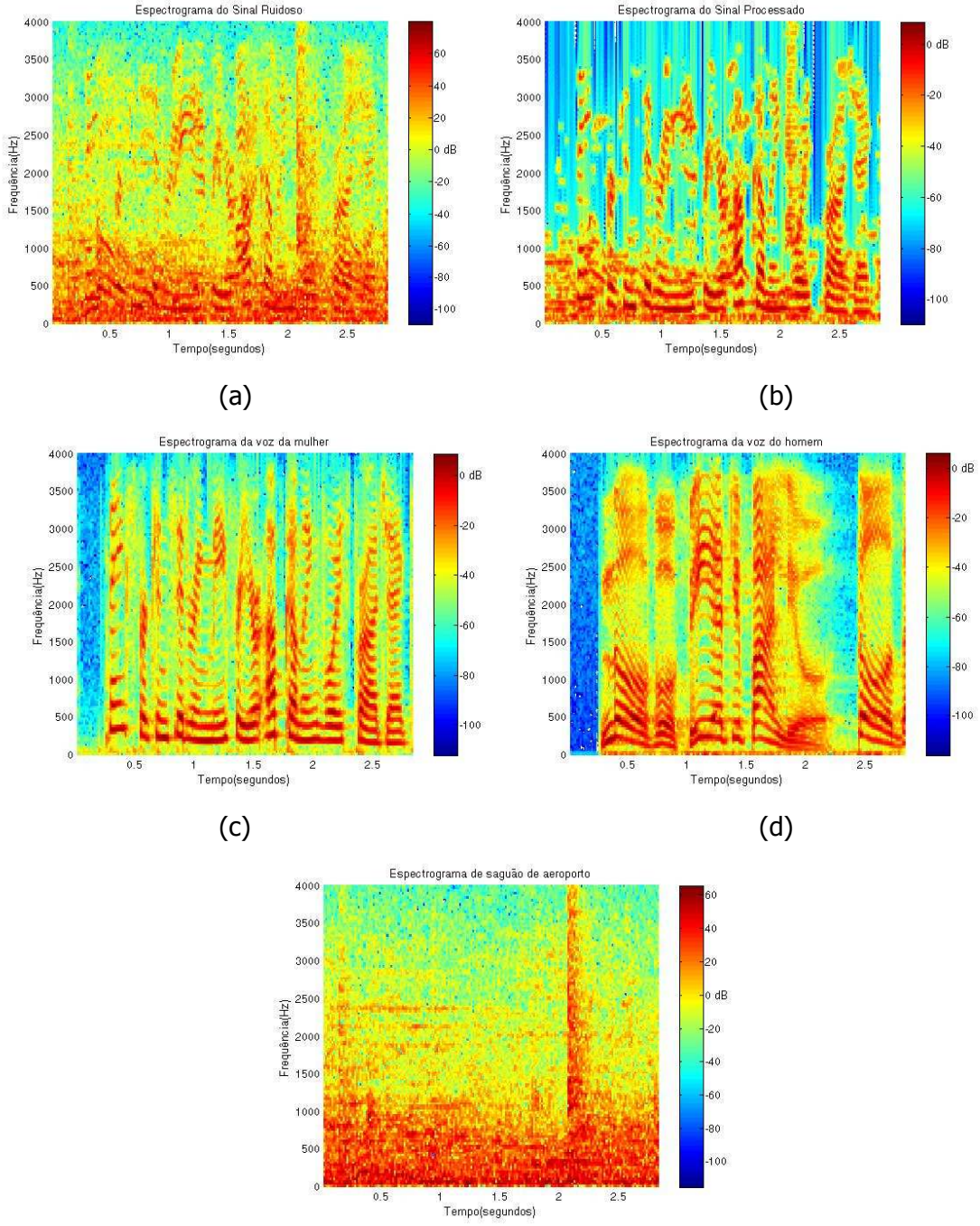


Figura 22: Identificação de fontes com um *array* de 4 microfones + subração espectral: (a) Sinal ruidoso com homem + mulher + saguão de aeroporto; (b) Sinal processado, priorizando a mulher; (c) Original voz da mulher; (d) Original voz do homem; (e) Saguão de aeroporto.

5 Conclusões e Trabalhos Futuros

Verificamos que este trabalho resultou em algoritmos com possibilidades reais de implementação em um mercado com necessidades de boas soluções. Obtivemos informações, através da literatura e de pesquisadores da área, que grandes empresas no exterior investem pesado neste mercado: o mercado de processamento de áudio para teleconferência, escutas em ambientes (identificação de fontes) e comunicação em automobilismo competitivo.

Observamos alguma dificuldade no desenvolvimento do trabalho em obter informações específicas dos parâmetros dos algoritmos (como as constantes alfas e betas dos algoritmos recursivos), uma vez que estes ainda são material de estudo e, até mesmo, de patentes na área de processamento de sinais. Este problema foi contornado pelo conhecimento e rede de contatos da orientadora deste trabalho.

Uma continuação deste trabalho seria a implementação de filtros fracionários no processamento do *array*, para diminuição da adição de ruídos de processamento.

Também poderíamos projetar um pós-processamento para a remoção do ruído musical introduzido pela Modificação Espectral.

Referência Bibliográfica

[1] S. F. Boll, "Suppression of acoustic noise in speech using spectral subtraction", IEEE trans. Acoust., Speech, Signal processing, vol. ASSP-29, pp. 113-120, April 1979.

[2] Y. Kaneda and J. Ohga, "Adaptive microphone-array system for noise reduction", IEEE Trans. Acoust., Speech, Signal processing, vol. ASSP-34, pp. 1391-1400, Sep. 1986

[3] B. Widrow, J. R. Glover, J. M. McCool, J. Kaunitz, C. S. Williams, R. H. Hearn, J. R. Zeidler, E. Dong, and R. C. Goodlin, " Adaptive noise cancelling: principles and applications", Proc. IEEE, vol. 63, pp. 1692-1975, Dec. 1975.

[4] Jingdong Chen, Yiteng (Arden) Huang, and Jacob Benesty, "Filtering Techniques for Noise Reduction and Speech Enhancement", Bell Laboratories, Lucent Technologies.

[5] BURDIC, WILLIAM S., Underwater Acoustic System Analysis, 2 ed. Englewood Cliffs, New Jersey, Prentice-Hall, 1991.

[6] S.V. Vaseghi, Advanced digital Signal Processing and Noise Reduction. John Willey & Sons, West Sussex, England, 2000.

APÊNDICE – ALGORITMOS

Este apêndice apresenta os códigos dos algoritmos implementados para uso no software MatLab.

```
1- Subtração espectral:
clc;            %Limpa a tela
clear all;      %Deleta as variaveis ja existentes
close all;      %Fecha todas as janelas abertas

echo on;
% @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
% Professora Marianne
% Fabricio Faria Santana
% DRE 099203093
% Parametric Spectral Subtraction final
% VERSAO DO MATLAB: 7.1.0.246(R14) SERVICE PACK 3
% @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
echo off;

% Carregando o arquivo wav
[SIGNAL,FS,NBITS]=wavread('mulher');
var=1e-3;
sd=sqrt(var);
% Adicionando ruido
SIGNAL = SIGNAL + sd*[randn(1,length(SIGNAL))]';

% Definindo o tamanho dos blocos de amostras
BlockSize = 256;

% Definindo os alphas, b e n
alphaa = exp(-1/(4*FS/(BlockSize/2)));
alphad = exp(-1/(1*FS/(BlockSize/2)));
b = 2;
n = 1;

% Calculando o numero de blocos existentes no arquivo wav
```

```
Nb = floor(length(SIGNAL)/BlockSize);
% Inicializando o vetor de erro V e o vetor de saida limpo Sout
Velevb = (sd*sqrt(BlockSize))^b*(ones(1,BlockSize))';
Sout = [];
janela = hann(BlockSize);
aux1 = (zeros(1,BlockSize))';
aux2 = aux1;
aux3 = aux1;

for i = 1:(Nb*2-1)
    Xelevb           =           (abs(fft(SIGNAL((i-1)*BlockSize/2           +
1:((i+1)*BlockSize/2)).*sqrt(janela)))).^b;
    PHI           =           (phase(fft(SIGNAL((i-1)*BlockSize/2           +
1:((i+1)*BlockSize/2)).*sqrt(janela))));

    for k=1:BlockSize
        if (Xelevb(k)) >= (Velevb(k))
            Velevb(k) = Velevb(k)*alphaa + Xelevb(k)*(1 - alphaa);
        else
            Velevb(k) = Velevb(k)*alphad + Xelevb(k)*(1 - alphad);
        end;
    end;

     Selevb = (Xelevb - Velevb.*n);
     aux2 = aux1;
     aux1 = real(ifft((Selevb.^(1/b)).*exp(j*PHI)));
     aux3                                                                    =
aux1(1:BlockSize/2).*sqrt(janela(1:BlockSize/2))+aux2(BlockSize/2+1:BlockSize)
.*sqrt(janela(BlockSize/2+1:BlockSize));
     Sout = [Sout; aux3];

end;

% Dando play no sinal original
wavplay(SIGNAL, FS);
% Dando play no sinal processado
wavplay(Sout, FS);
```

```
plot(SIGNAL(1:30000));
hold;
plot(Sout(1:30000),'r');
title('Subtração Espectral');
legend('Sinal Ruidoso', 'Sinal Processado');
xlabel('Amostras');
ylabel('Amplitude');
```

2- Filtro de Wiener Paramétrico:

```
clc;            %Limpa a tela
clear all;      %Deleta as variaveis ja existentes
close all;      %Fecha todas as janelas abertas

echo on;
% @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
% Professora Marianne
% Fabricio Faria Santana
% DRE 099203093
% Parametric Wiener Filtering final
% VERSAO DO MATLAB: 7.1.0.246(R14) SERVICE PACK 3
% @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
echo off;

% Carregando o arquivo wav
[SIGNAL,FS,NBITS]=wavread('mulher');
% Adicionando ruido
var=1e-3;
sd=sqrt(var);
SIGNAL = SIGNAL + sd*[randn(1,length(SIGNAL))]';

% Definindo o tamanho dos blocos de amostras
BlockSize = 256;

% Definindo os alphas, os bethas, b e n
alphaa = exp(-1/(4*FS/(BlockSize/2)));
alphad = exp(-1/(1*FS/(BlockSize/2)));
bethaa =exp(-1/(0.005*FS/(BlockSize/2)));
bethad = exp(-1/(0.01*FS/(BlockSize/2)));
```

```
b = 2;
n = 1;

% Calculando o numero de blocos existentes no arquivo wav
Nb = floor(length(SIGNAL)/BlockSize);
% Inicializando o vetor de erro V e o vetor de saida limpo Sout
Velevb = (sd*sqrt(BlockSize))^b*(ones(BlockSize,1));
Xbarra_t = zeros(BlockSize,1);
Sout = [];
janela = hann(BlockSize);
aux1 = (zeros(1,BlockSize))';
aux2 = aux1;
aux3 = aux1;

for i = 1:(Nb*2-1)
    X_t             =             abs(fft(SIGNAL((i-1)*BlockSize/2             +
1:((i+1)*BlockSize/2)).*sqrt(janela)));
    Xelevb = X_t.^b;
    PHI            =            (phase(fft(SIGNAL((i-1)*BlockSize/2            +
1:((i+1)*BlockSize/2)).*sqrt(janela))));

    for k=1:BlockSize
        if (Xelevb(k)) >= (Velevb(k))
            Velevb(k) = Velevb(k)*alphaa + Xelevb(k)*(1 - alphaa);
        else
            Velevb(k) = Velevb(k)*alphad + Xelevb(k)*(1 - alphad);
        end;
        if (X_t(k)) >= (Xbarra_t(k))
            Xbarra_t(k) = Xbarra_t(k)*bethaa + X_t(k)*(1 - bethaa);
        else
            Xbarra_t(k) = Xbarra_t(k)*bethad + X_t(k)*(1 - bethad);
        end;
    end
    E_N = Xbarra_t./(Velevb.^(1/b));

    Hpw = (1 - n*(1./(E_N.^b))).^(1/b);

    S = Hpw.*Xbarra_t;
```

```
    aux2 = aux1;
    aux1 = real(ifft((S).*exp(j*PHI)));
    aux3 =
aux1(1:BlockSize/2).*sqrt(janela(1:BlockSize/2))+aux2(BlockSize/2+1:BlockSize)
.*sqrt(janela(BlockSize/2+1:BlockSize));
    Sout = [Sout; aux3];

end;

% Dando play no sinal original
wavplay(SIGNAL, FS);
% Dando play no sinal processado
wavplay(Sout, FS);

plot(SIGNAL(1:30000));
hold;
plot(Sout(1:30000),'r');
title('Filtro de Wiener Paramétrico');
legend('Sinal Ruidoso', 'Sinal Processado');
xlabel('Amostras');
ylabel('Amplitude');
```

3- *Array* puro na remoção de ruído gaussiano:

```
clc;            %Limpa a tela
clear all;      %Deleta as variaveis ja existentes
close all;      %Fecha todas as janelas abertas

echo on;
% @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
% Professora Marianne
% Fabricio Faria Santana
% DRE 099203093
% Array de microfones gaussiano >> Remoção de ruído gaussiano com array
% VERSAO DO MATLAB: 7.1.0.246(R14) SERVICE PACK 3
% @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
echo off;

% Carregando os arquivos wav
```

```
[MULHER,FS,NBITS]=wavread('mulher');

var=1e-3;
sd=sqrt(var);
RUIDO = sd*[randn(1,length(MULHER))]';


% Criando array de microfones com ruído
% Considerando: velocidade do som igual a 340m/s.
% Array posicionado no parabrisas do querido Fusca, em frente ao
% passageiro, a mulher, que fica com angulo de incidencia de 90, sendo que
% o homem, o motorista, fica com angulo de incidencia de 20 graus.
% A distancia d entre os microfones sera de 0.05m.

Vsom = 340;
M = 10; % Numero de microfones

thetaM = 90; % Angulo de incidencia para mulher
cos_thetaM = cos(pi*thetaM/180);

d = 0.05;

% Para mulher:
delay_unitarioM = d*cos_thetaM/Vsom;
delay_amostrasM = floor(delay_unitarioM*FS)+1;

% Emulando sons captados pelos microfones:
MIC_1_mulher = MULHER;
MIC_2_mulher = cat(1, (zeros(1,delay_amostrasM*1))', MULHER(1:length(MULHER)-
delay_amostrasM*1));
MIC_3_mulher = cat(1, (zeros(1,delay_amostrasM*2))', MULHER(1:length(MULHER)-
delay_amostrasM*2));
MIC_4_mulher = cat(1, (zeros(1,delay_amostrasM*3))', MULHER(1:length(MULHER)-
delay_amostrasM*3));
MIC_5_mulher = cat(1, (zeros(1,delay_amostrasM*4))', MULHER(1:length(MULHER)-
delay_amostrasM*4));
MIC_6_mulher = cat(1, (zeros(1,delay_amostrasM*5))', MULHER(1:length(MULHER)-
delay_amostrasM*5));
```

```
MIC_7_mulher = cat(1, (zeros(1,delay_amostrasM*6))', MULHER(1:length(MULHER)-
delay_amostrasM*6));
MIC_8_mulher = cat(1, (zeros(1,delay_amostrasM*7))', MULHER(1:length(MULHER)-
delay_amostrasM*7));
MIC_9_mulher = cat(1, (zeros(1,delay_amostrasM*8))', MULHER(1:length(MULHER)-
delay_amostrasM*8));
MIC_10_mulher = cat(1, (zeros(1,delay_amostrasM*9))', MULHER(1:length(MULHER)-
delay_amostrasM*9));


MIC_1 = MIC_1_mulher + RUIDO(1:length(MULHER));
MIC_2 = MIC_2_mulher + RUIDO(1:length(MULHER));
MIC_3 = MIC_3_mulher + RUIDO(1:length(MULHER));
MIC_4 = MIC_4_mulher + RUIDO(1:length(MULHER));
MIC_5 = MIC_5_mulher + RUIDO(1:length(MULHER));
MIC_6 = MIC_6_mulher + RUIDO(1:length(MULHER));
MIC_7 = MIC_7_mulher + RUIDO(1:length(MULHER));
MIC_8 = MIC_8_mulher + RUIDO(1:length(MULHER));
MIC_9 = MIC_9_mulher + RUIDO(1:length(MULHER));
MIC_10 = MIC_10_mulher + RUIDO(1:length(MULHER));

% Corrigindo o delay dos microfones para identificacao da voz da mulher:
MIC_1 = MIC_1;
MIC_2                                                                    =
cat(1,MIC_2(delay_amostrasM*1+1:length(MIC_2)),(zeros(1,delay_amostrasM*1))');
MIC_3                                                                    =
cat(1,MIC_3(delay_amostrasM*2+1:length(MIC_3)),(zeros(1,delay_amostrasM*2))');
MIC_4                                                                    =
cat(1,MIC_4(delay_amostrasM*3+1:length(MIC_4)),(zeros(1,delay_amostrasM*3))');
MIC_5                                                                    =
cat(1,MIC_5(delay_amostrasM*4+1:length(MIC_5)),(zeros(1,delay_amostrasM*4))');
MIC_6                                                                    =
cat(1,MIC_6(delay_amostrasM*5+1:length(MIC_6)),(zeros(1,delay_amostrasM*5))');
MIC_7                                                                    =
cat(1,MIC_7(delay_amostrasM*6+1:length(MIC_7)),(zeros(1,delay_amostrasM*6))');
MIC_8                                                                    =
cat(1,MIC_8(delay_amostrasM*7+1:length(MIC_8)),(zeros(1,delay_amostrasM*7))');
```

```
MIC_9                                                                    =
cat(1,MIC_9(delay_amostrasM*8+1:length(MIC_9)),(zeros(1,delay_amostrasM*8))');
MIC_10                                                                   =
cat(1,MIC_10(delay_amostrasM*9+1:length(MIC_10)),(zeros(1,delay_amostrasM*9))'
);

MEDIA_MIC = (MIC_1+MIC_2+MIC_3+MIC_4+MIC_5+MIC_6+MIC_7+MIC_8+MIC_9+MIC_10)/M;
% MEDIA_MIC = (MIC_1+MIC_2+MIC_3+MIC_4)/M;

wavplay(MIC_1, FS);
wavplay(MEDIA_MIC, FS);

plot(MIC_1(1:30000));
hold;
plot(MEDIA_MIC(1:30000),'r');
title('Array de microfones com 10 microfones');
legend('Sinal Ruidoso', 'Sinal Processado');
xlabel('Amostras');
ylabel('Amplitude');


wavwrite(MEDIA_MIC, FS, 16, 'array_puro_mulher_10MICS');
```

4- *Array* + Subtração Espectral na remoção de ruído e identificação de fontes:

```
clc;            %Limpa a tela
clear all;      %Deleta as variaveis ja existentes
close all;      %Fecha todas as janelas abertas

echo on;
% @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
% Professora Marianne
% Fabricio Faria Santana
% DRE 099203093
% Array de microfones aeroporto >> Identificacao de fontes em um saguao de
% aeroporto com o uso da subtracao espectral uma unica vez
% VERSAO DO MATLAB: 7.1.0.246(R14) SERVICE PACK 3
% @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
echo off;
```

```
% Carregando os arquivos wav
[MULHER,FS,NBITS]=wavread('mulher');
[HOMEM,FS,NBITS]=wavread('homem');
[RUIDO,FS,NBITS]=wavread('saguao_aeroporto');
RUIDO = RUIDO*3;

if length(MULHER)<=length(HOMEM)
    HOMEM = HOMEM(1:length(MULHER));
else
    MULHER = MULHER(1:length(HOMEM));
end;

var=1e-3;
sd=sqrt(var);

% Criando array de microfones com ruído
% Considerando: angulo de 30 graus e velocidade do som igual a 340 m/s.
% A distancia d entre os microfones tera relacao direta com a precisao do
% sistema para identificacao de fontes e com a minima frequencia emitida
% pelas fontes.

Fmin = 350;
Vsom = 340;
M = 4; % Numero de microfones

thetaM = 20; % Angulo de incidencia para mulher
cos_thetaM = cos(pi*thetaM/180);

thetaH = 35; % Angulo de incidencia para homem
cos_thetaH = cos(pi*thetaH/180);

d = (1/(2*Fmin))*Vsom/abs(cos_thetaM - cos_thetaH);

% Para mulher:
delay_unitarioM = d*cos_thetaM/Vsom;
delay_amostrasM = floor(delay_unitarioM*FS)+1;
% Para homem:
```

```
delay_unitarioH = d*cos_thetaH/Vsom;
delay_amostrasH = floor(delay_unitarioH*FS)+1;

% Emulando sons captados pelos microfones:
MIC_1_mulher = MULHER;
MIC_2_mulher = cat(1, (zeros(1,delay_amostrasM*1))', MULHER(1:length(MULHER)-
delay_amostrasM*1));
MIC_3_mulher = cat(1, (zeros(1,delay_amostrasM*2))', MULHER(1:length(MULHER)-
delay_amostrasM*2));
MIC_4_mulher = cat(1, (zeros(1,delay_amostrasM*3))', MULHER(1:length(MULHER)-
delay_amostrasM*3));
%       MIC_5_mulher       =       cat(1,       (zeros(1,delay_amostrasM*4))',
MULHER(1:length(MULHER)-delay_amostrasM*4));
%       MIC_6_mulher       =       cat(1,       (zeros(1,delay_amostrasM*5))',
MULHER(1:length(MULHER)-delay_amostrasM*5));
%       MIC_7_mulher       =       cat(1,       (zeros(1,delay_amostrasM*6))',
MULHER(1:length(MULHER)-delay_amostrasM*6));
%       MIC_8_mulher       =       cat(1,       (zeros(1,delay_amostrasM*7))',
MULHER(1:length(MULHER)-delay_amostrasM*7));
%       MIC_9_mulher       =       cat(1,       (zeros(1,delay_amostrasM*8))',
MULHER(1:length(MULHER)-delay_amostrasM*8));
%      MIC_10_mulher       =       cat(1,       (zeros(1,delay_amostrasM*9))',
MULHER(1:length(MULHER)-delay_amostrasM*9));

MIC_1_homem = HOMEM;
MIC_2_homem  =  cat(1,  (zeros(1,delay_amostrasH*1))',  HOMEM(1:length(HOMEM)-
delay_amostrasH*1));
MIC_3_homem  =  cat(1,  (zeros(1,delay_amostrasH*2))',  HOMEM(1:length(HOMEM)-
delay_amostrasH*2));
MIC_4_homem  =  cat(1,  (zeros(1,delay_amostrasH*3))',  HOMEM(1:length(HOMEM)-
delay_amostrasH*3));
% MIC_5_homem = cat(1, (zeros(1,delay_amostrasH*4))', HOMEM(1:length(HOMEM)-
delay_amostrasH*4));
% MIC_6_homem = cat(1, (zeros(1,delay_amostrasH*5))', HOMEM(1:length(HOMEM)-
delay_amostrasH*5));
% MIC_7_homem = cat(1, (zeros(1,delay_amostrasH*6))', HOMEM(1:length(HOMEM)-
delay_amostrasH*6));
```

```
% MIC_8_homem = cat(1, (zeros(1,delay_amostrasH*7))', HOMEM(1:length(HOMEM)-
delay_amostrasH*7));
% MIC_9_homem = cat(1, (zeros(1,delay_amostrasH*8))', HOMEM(1:length(HOMEM)-
delay_amostrasH*8));
% MIC_10_homem = cat(1, (zeros(1,delay_amostrasH*9))', HOMEM(1:length(HOMEM)-
delay_amostrasH*9));

MIC_1 = MIC_1_mulher + MIC_1_homem + RUIDO(1:length(MULHER));
MIC_2 = MIC_2_mulher + MIC_2_homem + RUIDO(1:length(MULHER));
MIC_3 = MIC_3_mulher + MIC_3_homem + RUIDO(1:length(MULHER));
MIC_4 = MIC_4_mulher + MIC_4_homem + RUIDO(1:length(MULHER));
% MIC_5 = MIC_5_mulher + MIC_5_homem + sd*[randn(1,length(MULHER))]';
% MIC_6 = MIC_6_mulher + MIC_6_homem + sd*[randn(1,length(MULHER))]';
% MIC_7 = MIC_7_mulher + MIC_7_homem + sd*[randn(1,length(MULHER))]';
% MIC_8 = MIC_8_mulher + MIC_8_homem + sd*[randn(1,length(MULHER))]';
% MIC_9 = MIC_9_mulher + MIC_9_homem + sd*[randn(1,length(MULHER))]';
% MIC_10 = MIC_10_mulher + MIC_10_homem + sd*[randn(1,length(MULHER))]';

% % Corrigindo o delay dos microfones para identificacao da voz do homem:
% MIC_1 = MIC_1;
%                                   MIC_2                                   =
cat(1,MIC_2(delay_amostrasH*1+1:length(MIC_2)),(zeros(1,delay_amostrasH*1))');
%                                   MIC_3                                   =
cat(1,MIC_3(delay_amostrasH*2+1:length(MIC_3)),(zeros(1,delay_amostrasH*2))');
%                                   MIC_4                                   =
cat(1,MIC_4(delay_amostrasH*3+1:length(MIC_4)),(zeros(1,delay_amostrasH*3))');
%                       %                       MIC_5                       =
cat(1,MIC_5(delay_amostrasH*4+1:length(MIC_5)),(zeros(1,delay_amostrasH*4))');
%                       %                       MIC_6                       =
cat(1,MIC_6(delay_amostrasH*5+1:length(MIC_6)),(zeros(1,delay_amostrasH*5))');
%                       %                       MIC_7                       =
cat(1,MIC_7(delay_amostrasH*6+1:length(MIC_7)),(zeros(1,delay_amostrasH*6))');
%                       %                       MIC_8                       =
cat(1,MIC_8(delay_amostrasH*7+1:length(MIC_8)),(zeros(1,delay_amostrasH*7))');
%                       %                       MIC_9                       =
cat(1,MIC_9(delay_amostrasH*8+1:length(MIC_9)),(zeros(1,delay_amostrasH*8))');
```

```
%                       %                       MIC_10                          =
cat(1,MIC_10(delay_amostrasH*9+1:length(MIC_10)),(zeros(1,delay_amostrasH*9))'
);

% Corrigindo o delay dos microfones para identificacao da voz da mulher:
MIC_1 = MIC_1;
MIC_2                                                                           =
cat(1,MIC_2(delay_amostrasM*1+1:length(MIC_2)),(zeros(1,delay_amostrasM*1))');
MIC_3                                                                           =
cat(1,MIC_3(delay_amostrasM*2+1:length(MIC_3)),(zeros(1,delay_amostrasM*2))');
MIC_4                                                                           =
cat(1,MIC_4(delay_amostrasM*3+1:length(MIC_4)),(zeros(1,delay_amostrasM*3))');
%                                   MIC_5                                       =
cat(1,MIC_5(delay_amostrasM*4+1:length(MIC_5)),(zeros(1,delay_amostrasM*4))');
%                                   MIC_6                                       =
cat(1,MIC_6(delay_amostrasM*5+1:length(MIC_6)),(zeros(1,delay_amostrasM*5))');
%                                   MIC_7                                       =
cat(1,MIC_7(delay_amostrasM*6+1:length(MIC_7)),(zeros(1,delay_amostrasM*6))');
%                                   MIC_8                                       =
cat(1,MIC_8(delay_amostrasM*7+1:length(MIC_8)),(zeros(1,delay_amostrasM*7))');
%                                   MIC_9                                       =
cat(1,MIC_9(delay_amostrasM*8+1:length(MIC_9)),(zeros(1,delay_amostrasM*8))');
%                                  MIC_10                                       =
cat(1,MIC_10(delay_amostrasM*9+1:length(MIC_10)),(zeros(1,delay_amostrasM*9))'
);

%                                 MEDIA_MIC                                     =
(MIC_1+MIC_2+MIC_3+MIC_4+MIC_5+MIC_6+MIC_7+MIC_8+MIC_9+MIC_10)/M;
MEDIA_MIC = (MIC_1+MIC_2+MIC_3+MIC_4)/M;

% Definindo o tamanho dos blocos de amostras
BlockSize = 256;

% Definindo os alphas, b e n
alphaa = exp(-1/(4*FS/(BlockSize/2)));
alphad = exp(-1/(1*FS/(BlockSize/2)));
b = 2;
n = 1;
```

```
% Calculando o numero de blocos existentes no arquivo wav
Nb = floor(length(MEDIA_MIC)/BlockSize);
% Inicializando o vetor de erro V e o vetor de saida limpo Sout
Velevb = (sd*sqrt(BlockSize))^b*(ones(1,BlockSize))';
Sout = [];
janela = hann(BlockSize);
aux1 = (zeros(1,BlockSize))';
aux2 = aux1;
aux3 = aux1;

for i = 1:(Nb*2-1)
    Xelevb          =          (abs(fft(MEDIA_MIC((i-1)*BlockSize/2          +
1:((i+1)*BlockSize/2)).*sqrt(janela)))).^b;
    PHI          =          (phase(fft(MEDIA_MIC((i-1)*BlockSize/2          +
1:((i+1)*BlockSize/2)).*sqrt(janela))));

    for k=1:BlockSize
        if (Xelevb(k)) >= (Velevb(k))
            Velevb(k) = Velevb(k)*alphaa + Xelevb(k)*(1 - alphaa);
        else
            Velevb(k) = Velevb(k)*alphad + Xelevb(k)*(1 - alphad);
        end;
    end;

     Selevb = (Xelevb - Velevb.*n);
     aux2 = aux1;
     aux1 = real(ifft((Selevb.^(1/b)).*exp(j*PHI)));
     aux3                                                                    =
aux1(1:BlockSize/2).*sqrt(janela(1:BlockSize/2))+aux2(BlockSize/2+1:BlockSize)
.*sqrt(janela(BlockSize/2+1:BlockSize));
     Sout = [Sout; aux3];

end;

% Dando play no sinal original
wavplay(MIC_1, FS);
% Dando play no sinal processado
```

```
wavplay(Sout, FS);

plot(MIC_1(1:30000));
hold;
plot(Sout(1:30000),'r');
title('Array de microfones + Subtração Espectral');
legend('Sinal Ruidoso', 'Sinal Processado');
xlabel('Amostras');
ylabel('Amplitude');

% wavwrite(MIC_1, FS, 16, 'mulher+homem+aeroporto');
```