Sobre a utilização de filtragem adaptativa na melhoria de sinal de voz mediante a eliminação de trechos gravados a partir de áudio conhecido

José Antonio Apolinário Jr., Dirceu Gonzaga da Silva e Antonio César Morant Braid

***Resumo*— Filtragem adaptativa é uma técnica de processamento digital de sinais que tem sido usada para a melhoria de um sinal de voz que foi corrompido por um ruído aditivo indesejado; para sua implementação, freqüentemente dispõe-se de um segundo sinal correlacionado com o ruído e não correlacionado com o sinal de voz desejado. Tal sinal normalmente é proveniente de um segundo microfone que consegue captar a fonte de ruído sem gravar o sinal de interesse. Em alguns casos, um perito criminal pode se deparar com uma gravação cujo ruído indesejável é proveniente de uma música transmitida, por exemplo, por uma estação de rádio ou do áudio de um canal de televisão com o volume alto. Assumindo que temos o sinal interferidor original, música ou áudio de uma novela ou programa de televisão, este artigo aborda como utilizar um filtro adaptativo para a eliminação desta interferência, seus detalhes de emprego, suas limitações e seus resultados.**

***Palavras-Chave*— Perícia de voz, filtragem adaptativa, *speech signal enhancement*.**

I. Introdução

Um perito criminal, por vezes, pode se deparar com a seguinte situação: receber para perícia fonética um sinal de voz contaminado com um ruído aditivo, eventualmente até com maior intensidade que o original, proveniente de um alto-falante que reproduz uma música de um CD, DVD ou de um rádio, ou mesmo um sinal de áudio de qualquer programação da televisão. Em outras palavras, o sinal que contaminou a informação de interesse é conhecido (assumimos que o perito tem o respectivo CD, DVD ou uma cópia do programa de rádio ou de TV) mas, como o primeiro passou por todos os estágios de uma transmissão e, principalmente, pelo caminho acústico da caixa de som até o microfone, ele não pode ser simplesmente removido por mera subtração.

Para esta tarefa de eliminar um ruído, dispondo de um sinal fortemente correlacionado com ele, pode-se usar um filtro adaptativo. Tal princípio tem sido muito utilizado para o cancelamento de ruídos de sinais estacionários; como um exemplo, podemos citar o cancelador de ruído implementado em modernos *headphones* o qual é bastante eficiente para uma viagem de avião onde o ruído constante, de baixa frequência e quase estacionário das turbinas pode ser fortemente atenuado.

José A. Apolinário Jr¸ é professor do Programa de Pós-Graduação em Engenharia Elétrica do Instituto Militar de Engenharia (IME), Rio de Janeiro, RJ. Dirceu G. da Silva é diretor da InnoVox Processamento de Áudio e Voz, Rio de Janeiro, RJ. César Braid é perito criminal do Instituto de Criminalística Afrânio Peixoto (ICAP), do Departamento de Polícia Técnica do Estado da Bahia, Salvador, BA. E-mails: apolin@ime.eb.br, dirceu@innovoxtelecom.com.br e cesarbraid@hotmail.com

Este artigo aplica esta técnica conhecida como *speech enhancement* para o caso da perícia fonética e aborda suas principais características e limitações sem entrar em excessivos detalhes teóricos mas tendo como meta uma orientação ao perito em como utilizar a filtragem e efetivamente obter os melhores resultados possíveis. A organização da versão final deste trabalho é a que segue. A Seção II apresenta um pequeno resumo sobre os principais conceitos usados em filtragem adaptativa e em particular na sua aplicação ao problema de melhoria de voz. Na Seção III, é feita uma descrição do problema e daqueles parâmetros que devem ser fornecidos ao filtro adaptativo para que este consiga oferecer um bom resultado; esta seção também apresenta alguns detalhes de como foi implementado o programa de melhoria de sinal de voz. A Seção IV apresenta os resultados das simulações realizadas, com detalhes práticos da utilização do programa feito para esta aplicação, bem como uma breve descrição de um caso real. As conclusões deste trabalho são apresentadas na Seção V.

II. Filtragem adaptativa aplicada à melhoria de sinal

Um filtro adaptativo pode ser definido com um filtro digital que auto ajusta seus coeficientes de modo a minimizar uma função custo baseada num sinal de erro [1]. Este erro corresponde à diferença entre um sinal de referência e a saída do próprio filtro adaptativo. Um filtro adaptativo em sua configuração básica pode ser representado como na Fig. 1.

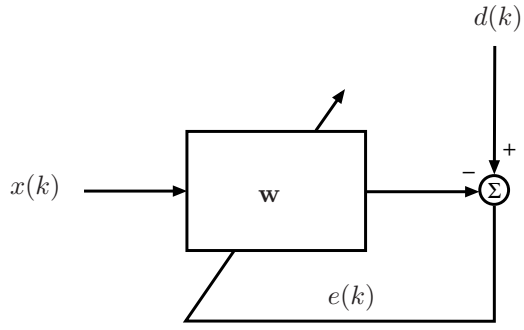

Fig. 1. Configuração básica de um filtro adaptativo: $x(k)$ é o sinal de entrada, $d(k)$ é o sinal de referência usado para produzir, com a saída do filtro, um erro $e(k)$ que é empregado na atualização do vetor de coeficientes $\mathbf{w}$ do filtro.

A cada instante de tempo k (note que todos os sinais foram digitalizados com uma certa frequência de amostragem f_s

e que o instante de tempo discreto k se relaciona com o tempo por meio da equação $t = kT$, T sendo o período de amostragem que corresponde ao inverso de f_s), os coeficientes de um filtro adaptativo são ajustados de acordo com um algoritmo que usa o erro entre a referência e sua saída para realizar tal atualização. Nesta figura, os coeficientes do filtro são representados pelo vetor $\mathbf{w}$, a entrada (que no nosso caso particular corresponderá a um sinal correlacionado à interferência) usualmente é representada por $x(n)$ e a referência (que no nosso caso corresponde à soma do sinal de interesse com o ruído que queremos eliminar) por $d(k)$. Desta forma, o sinal de erro é dado por $e(k) = d(k) - y(k)$ onde $y(k)$ é a saída dada por $\mathbf{w}^{\mathrm{T}}(k-1)\mathbf{x}(k)$, com o vetor de coeficientes computado no instante anterior–pois iremos usar este erro para obter o vetor atual $\mathbf{w}(k)$–e o vetor sinal de entrada $\mathbf{x}(k)$ formado pela entrada $x(k)$ e suas N últimas amostras anteriores, ou seja, $\mathbf{x}(k) = [x(k)\, x(k-1) \cdots x(k-N)]^T$, N é conhecido como a ordem do filtro adaptativo (observe que seu número de coeficientes é dado por $N+1$).

Existe uma grande variedade de algoritmos adaptativos e escolher aquele que é mais adequado à aplicação pode não ser uma tarefa fácil. Não é nossa intenção neste artigo, pela própria falta de espaço e por considerarmos fora do escopo desta apresentação, discutir detalhes específicos do filtro adaptativo utilizado. Contudo, de maneira geral, podemos afirmar que um algoritmo que converge para a solução ótima de maneira mais rápida requer uma maior complexidade computacional que pode ser traduzida na prática pelo maior tempo de processamento. Por outro lado, tendo em vista que uma gravação é usualmente feita num ambiente dinâmico, onde as pessoas se movimentam (trata-se de um sistema não-estacionário), precisamos usar um algoritmo que apresente uma rápida velocidade de convergência. Caso contrário, usando um algoritmo lento, o sistema que ele estava tentando modelar já teria, possivelmente antes da convergência, se modificado e o resultado seria certamente bem mais modesto.

Com esta breve discussão, já podemos apresentar duas características básicas que um projetista deve se preocupar para fazer o filtro adaptativo funcionar a contento nesta aplicação: estimar a ordem (N) do filtro e determinar qual algoritmo utilizar.

A Fig. 2 apresenta um filtro adaptativo usado especificamente no problema em tela, o cancelamento de um ruído *conhecido*. Observe neste caso que, se $\mathbf{w}$ é computado tal que $y(k) \approx n(k)$, o próprio erro de estimação corresponderia aproximadamente ao sinal de voz limpo, isto é, $e(k) \approx s(k)$. Contudo, como sempre teremos, além do ruído aditivo $n(k)$ outros ruídos de diversas fontes, podemos representar esta saída do processamento por $\hat{s}(k)$, uma estimativa de $s(k)$ que seria na verdade $s(k) + r(k)$ caso houvesse uma modelagem perfeita. O ruído $r(k)$ pode ser, por exemplo, um aparelho de ar-condicionado ligado.

Em aplicações de melhoria de áudio com dois microfones (um para o sinal de voz corrompido por um ruído e outro para um sinal correlacionado a este ruído), ambos são gerados simultaneamente. Isto é o caso, por exemplo, de um ruído de avião sendo cancelado no microfone que transmite o sinal de voz do piloto para uma torre de comando; isto contudo não

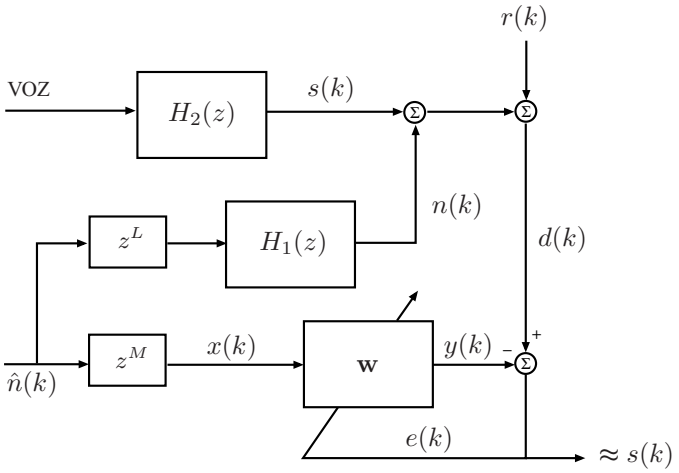

Fig. 2. Diagrama de blocos do filtro adaptativo usado para o cancelamento de ruído: dispomos do sinal de voz corrompido, $d(k)$, e da fonte da interferência, o sinal $\hat{n}(k)$, que pode ser uma música de um CD ou o áudio de um programa de TV.

ocorre no problema do perito em fonética forense: ele dispõe do sinal $\hat{n}(k)$ (sinal interferidor original, i.e., uma música de um DVD ou o áudio de um programa de TV) mas não sabe exatamente como alinhar temporalmente os dois sinais. Em outras palavras, não conhece o sincronismo entre os sinais e não sabe *a priori* quantas amostras retirar do início do sinal para que a tarefa do filtro adaptativo seja facilitada. Levando em conta que um filtro digital (adaptativo ou não) só consegue atrasar um sinal (sendo por isto conhecido como causal) e nunca adiantá-lo (ou antecipá-lo como um filtro anti-causal), se não recortarmos um número de amostras suficiente, o filtro não conseguirá eliminar o ruído por não conseguir alinhar temporalmente os sinais (ele, neste caso, necessitaria ser anti-causal). Se, por outro lado, cortamos muitas amostras, o filtro tenderá a gastar várias de suas amostras iniciais somente para a tarefa de retardar o sinal corretamente; o problema neste caso, seria a necessidade de aumentar muito a ordem do filtro tornando, por vezes, impraticável sua utilização (ele demoraria muito tempo para apresentar um resultado).

Na seção seguinte, estas três características do filtro adaptativo (ordem, algoritmo e sincronismo do sinais) serão abordadas, levando-se em conta um ambiente fictício onde conhecemos suas características, principalmente sua geometria. No caso real, tais dimensões não são conhecidas e devemos, de alguma maneira, estimar o sincronismo: para isto, assumimos uma ordem que seja viável para uma ampla gama de ambientes e usamos um algoritmo que seja rápido o suficiente para acompanhar a velocidade de variação do caminho acústico do sinal gravado.

III. Desenvolvendo o software *IMECleanSpeech*

Esta seção aborda o desenvolvimento de um software que, mediante uma correta entrada de parâmetros, possibilitará uma melhoria do sinal de voz corrompido por áudio previamente conhecido. Da Fig. 2, pode-se perceber que o filtro adaptativo, ao eliminar a correlação existente entre os sinais $d(k)$ e $x(k)$, deverá convergir a um vetor de coeficientes $\mathbf{w}$ tal que $w(k+M) = h_1(k+L)$, de tal forma que o erro de estimação

$e(k)$ seja, aproximadamente, o próprio sinal de interesse $s(k)$. Tal sinal, $s(k)$, em realidade corresponde à convolução do sinal de voz do locutor que está sendo gravado num dado instante com a resposta ao impulso do ambiente, considerando-se o percurso entre o locutor e o microfone, na figura representado no domínio da Transformada $\mathcal{Z}$ por $H_2(z)$. Não se pretende, contudo, eliminar esta reverberação do sinal de voz mas sim considerarmos este sinal $s(k)$ o nosso objetivo. Ao sinal de interesse foi somado o ruído $n(k)$ que corresponde a um sinal conhecido $\hat{n}(k)$ (uma música ou qualquer áudio conhecido, reproduzido no ambiente por um alto-falante assumido de boa qualidade). Observe na figura que, ao iniciar a gravação, assumimos que já havia passado um certo tempo (o sinal foi recortado em L amostras), ou seja, a música já havia iniciado. Tal situação corresponde ao que normalmente encontramos na prática. Desta forma, $\hat{n}(k+L)$, o sinal conhecido mas com um retardo desconhecido, passa por um outro caminho acústico (do alto-falante ao microfone), gerando $n(k)$ que pode ser representado por $\hat{n}(k+L) * h_1(k)$. Finalmente, o sinal que temos gravado, indicado por $d(k)$, corresponde ao sinal de interesse somado ao sinal conhecido deslocado no tempo e filtrado, e ainda somado a um segundo ruído ambiente, também conhecido como ruído *de observação*, $r(k)$.

Note que temos em mãos uma gravação $d(k)$ e o sinal conhecido $\hat{n}(k)$ o qual desejamos remover, ficando com $e(k) \approx s(k)$ (o ruído ambiente também permanecerá). Estamos, ao usar um filtro adaptativo linear, considerando o caminho do *áudio conhecido* correspondente a um filtro linear (desconsiderando não-linearidades do modo como foi transmitido[1]), possivelmente variante no tempo caso haja qualquer movimento do locutor ou do ambiente.

Na Fig. 2, representamos como M o valor que o sinal conhecido $\hat{n}(k)$ deverá ser recortado de modo a conseguirmos um bom resultado em sua eliminação. Isto deve ser feito sem sabermos *a priori* o valor correto de L. O que faremos é computar a correlação entre o sinais que temos, $\hat{n}(k)$ e $d(k)$ (que representaremos por $r_{\hat{n}d}(\tau)$): quando tal correlação for máxima, o valor de τ correspondente será igual a L mais o retardo de grupo (médio) do filtro $H_1(k)$.

Antes de prosseguirmos, realizamos um pequeno experimento no qual, durante a reprodução de uma música proveniente de um DVD com um alto-falante de boa qualidade, gravou-se uma voz de um locutor masculino, usando a frequência de amostragem de $44100Hz$. Neste ponto, convém ressaltar que usar uma frequência de amostragem tão alta é inconveniente pois os requerimentos computacionais para o processamento são muito altos e quase toda a informação relevante encontra-se, normalmente, nas frequências mais baixas. Desta forma, sugere-se a frequência de amostragem de $8kHz$ para o processamento. Caso os sinais estejam em frequências diferentes, devemos efetuar uma transformação de taxa de amostragem para que ambos fiquem em $8KHz$. Como tal transformação é corriqueira e muitos softwares de áudio a realizam, este assunto não será abordado.

Realizamos os testes iniciais em Matlab® e, para o exemplo descrito nesta seção, um trecho da curva de correlação cruzada, obtida basicamente como $\texttt{mean}(\hat{n}(k).*d(k-\tau))$ a qual foi implementada em Matlab® no domínio da frequência para termos maior eficiência e portanto maior rapidez, está mostrada na Fig. 3; nota-se claramente nesta figura um pico em um determinado valor de τ, em princípio, correspondente à soma de L com o *group delay* médio de $h_1(k)$.

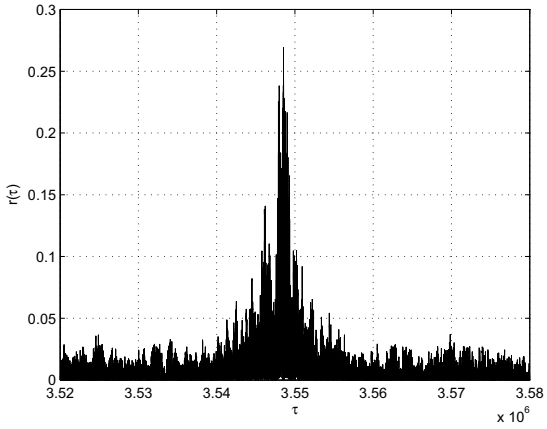

Fig. 3. Curva da correlação entre o áudio conhecido, $\hat{n}(k)$, e a gravação da voz corrompida, $d(k)$.

Com este valor de τ e uma estimativa do retardo de grupo de $h_1(k)$ correspondente ao caminho acústico do alto-falante ao microfone, conseguimos efetuar um recorte do sinal de áudio, ou seja, computar um valor de M que possibilite bons resultados. A estimativa do retardo de grupo de $h_1(k)$ não é uma tarefa simples. Neste trabalho, utilizamos para tal estimação um ambiente fictício de dimensões $3m \times 4m$ com $2.5m$ de altura para, utilizando uma rotina de geração de resposta ao impulso (*room impulse response*) [2], fazermos um histograma dos retardos de grupo e das ordens de tais respostas para diferentes pontos deste ambiente. Como resultado, tomamos um valor médio e fizemos as seguintes propostas de valores em função da frequência de amostragem:

$$N = ceil(0.1303 f_s) \quad (1)$$

$$\tau_{h_1} = ceil(0.0352 f_s - 66.88) \quad (2)$$

onde N é a ordem do filtro adaptativo e τ_{h_1} corresponde ao retardo de grupo médio estimado para o RIR (*room impulse response*) $h_1(k)$, do caminho acústico entre o alto-falante e o microfone. A função do Matlab® `ceil` arredonda o número para o inteiro mais próximo em direção ao infinito.

A escolha do valor de M, N e τ tem um grande impacto no resultado. Levando-se em consideração que o retardo de grupo não é constante (ele varia em função da frequência), usarmos simplesmente o valor de τ obtido (em princípio igual a $L+\tau_{h_1}$ pode ocasionar uma perda de desempenho pois pode requerer que o filtro adaptativo seja não-causal (antecipativo); logo, para garantir tal situação, usamos um valor de M igual a τ da

[1]Esta consideração é importante pois, caso contrário, caso haja algum tipo de não-linearidade (volume exageradamente alto, por exemplo, levando à distorção harmônica do sinal $n(k)$), o filtro adaptativo linear como assumido aqui não será capaz de eliminar o ruído. A extensão desta técnica usando um filtro adaptativo não-linear para contornar tal limitação é assunto de pesquisa em andamento.

correlação máxima mais o valor τ_{h_1} estimado por (2). Espera-se, desta forma, ter-se um pouco de folga para poder garantir um filtro adaptativo causal sem aumentar muito a sua ordem. O valor de N, por outro lado, se pequeno, pode causar uma modelagem de baixa qualidade (*undermodeling*) mas se for muito grande, poderá tornar o filtro adaptativo muito lento ou mesmo, dependendo do algoritmo utilizado, inviável. No caso do exemplo apresentado nesta seção, tivemos como resultado τ_{h_1} estimado em 1486, $M = \tau + \tau_{h_1} = 3015053$ e $N = 5747$.

Com os valores usados neste experimento, em ambiente Matlab®, o processamento usando o algoritmo NLMS (Normalized Least Mean Squares) [1] levou cerca de 128 segundos num processador AMD Turion64 de $1.8GHz$. O resultado melhorou claramente a inteligibilidade por retirar uma parcela considerável da música; contudo, tendo em vista a conhecida baixa velocidade de convergência do algoritmo NLMS e a não estacionariedade do ambiente, espera-se um resultado ainda melhor quando usarmos um algoritmo da família RLS.

IV. Resultados das Simulações

Conforme visto na seção anterior, o algoritmo NLMS possui uma baixa complexidade computacional mas também não possibilita uma alta velocidade de convergência; buscaremos a seguir alternativas mais rápidas a este algoritmo. Antes de partirmos para um algoritmo de rápida convergência mas com alta complexidade computacional (os da família RLS), implementamos uma segunda alternativa, o algoritmo BNDR-LMS [1], o qual é mais rápido que o algoritmo NLMS mas ainda executável em tempo relativamente curto.

Para os algoritmos NLMS e BNDR-LMS, além dos parâmetros já mencionados, existe ainda aquele conhecido como *tamanho do passo* (do inglês *step-size*). Este parâmetro, conhecido pela letra grega μ, controla a velocidade de convergência e a proximidade que o vetor de coeficientes se aproxima do ótimo após a convergência (desajuste). Em termos gerais, quanto maior (mais próximo de 1) o valor de μ, mais rápida a convergência e maior o desajuste e vice-versa. As gravações deste experimento foram realizadas com uma leve não-estacionariedade (pouca movimentação do único locutor no ambiente) e precisamos ter uma idéia do valor de μ que seja adequado a esta aplicação: este valor não pode ser muito grande devido ao grande desajuste de algoritmos normalizados (como no caso do NLMS e do BNDR-LMS) e nem muito pequeno pois antes de convergir o ambiente já mudaria e o filtro tentaria acompanhar com uma baixa velocidade. É comum, pois, existir um valor ótimo de μ para cada aplicação. Para ilustrar isto, dois sinais foram gravados em separado, de maneira a podermos ter os sinais $n(k)$ e $s(k)$ isolados, e com isto o controle sobre a relação-sinal-ruído recebida no microfone. Com este tipo de experimento, podemos também medir objetivamente o desempenho do filtro adaptativo.

Para este experimento, baixamos a frequência de amostragem para $8KHz$ e ajustamos a relação-sinal-ruído da entrada para zero dB, ou seja, a potência do ruído sendo igual a do sinal:

$$SNR_{dB} = 10log\frac{\sigma_s^2}{\sigma_n^2} = 0 \qquad (3)$$

onde σ_s^2 é a variância do sinal $s(k)$ (energia do sinal) e σ_n^2 a variância de $n(k)$ (energia do ruído).

Para este sinais, variamos o valor de μ e medimos a relação-sinal-ruído após o processamento, ou seja, aquela medida entre o sinal original e o ruído no sinal de saída (obviamente, quanto maior o valor da SNR_{dB} melhor o desempenho do filtro adaptativo). Os resultados estão na Fig. 4 onde pode-se observar que os melhores desempenhos para os algoritmos NLMS e BNDR-LMS, para este experimento, ocorrem para valores de *step-sizes* de 0.035 e 0.015, respectivamente.

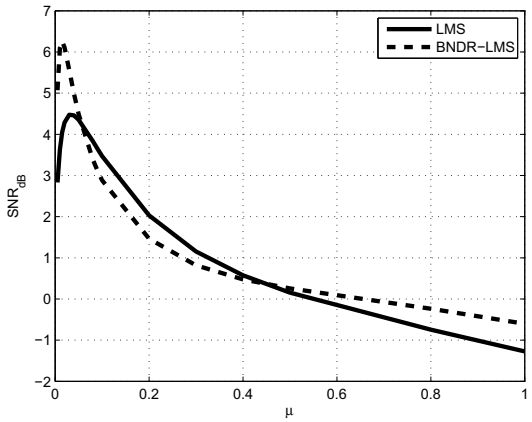

Fig. 4. Relação-sinal-ruído medida no sinal após processamento com os algoritmos NLMS e BNDR-LMS em função dos valores de *step-sizes*. Note que os valores relativamente baixos de μ que resultam nos picos de SNR sugerem que a planta (caminho acústico do sinal interferidor) é levemente não-estacionária; ambientes com não-estacionariedade mais acentuada demandariam valores mais elevados de μ.

Para implementarmos um algoritmo da família RLS, optou-se por uma versão rápida usando decomposição QR que, além de estável (o algoritmo RLS convencional não é estável), apresenta uma complexidade computacional da ordem de N multiplicações por iteração. Entretanto, em ambiente Matlab®, não logrou-se rodar em tempo razoável e com os mesmos recursos computacionais o exemplo anterior (mesmo baixando-se a frequência de amostragem de $44100Hz$ para $8KHz$). Para rodar os algoritmos denominados FQRD-RLS [3], [4] de modo a termos um melhor desempenho que os baseados em LMS (NLMS e BNDR-LMS), foi necessária a implementação em uma linguagem de alto nível (C) de maneira que fossem rápido o suficiente para rodar uma grande quantidade de dados com o filtro possuindo um elevado número de coeficientes.

Usando frequência de amostragem de $8KHz$, os tempos de processamento foram:

1) algoritmo NLMS (resultado pobre) rodando em Matlab: aproximadamente 6 segundos;
2) algoritmo BNDR-LMS (resultado médio) rodando em Matlab: aproximadamente 10 segundos; e
3) algoritmo FQRD-RLS (resultado bom) rodando em programa executável (C): aproximadamente 1 minuto.

Ressalta-se, para aqueles interessados na programação do algoritmo FQRD-RLS, que a restrição de rotações passivas

($\cos^2\theta + \sin^2\theta \leq 1$, para qualquer ângulo θ das rotações de Givens usadas) foi imposta na sua implementação em linguagem C. Tal condição foi necessária para evitar possível instabilidade típica de implementação em precisão finita.

Para verificarmos qual parâmetro λ (forgetting factor) do algoritmo RLS deveria ser empregado no algoritmo FQRD-RLS implementado, repetimos o experimento anterior (sinal corrompido com $SNR_{dB} = 0$) com $\lambda = 0.9999$, 0.99999, 0.999999 e 1; os resultados, em termos de SBR_{dB} após processamento, foram 7.4003, 9.0021, 9.0647 e 9.0682, respectivamente. Logo, apesar de um pequeno espaço amostral, os resultados sugerem um valor $\lambda = 1$ como o que apresenta um melhor desempenho.

Em seguida, para comparar o desempenho do algoritmo FQRD-RLS com os algoritmos NLMS e BNDR-LMS, rodamos (usando os melhores valores de μ já obtidos) os três algoritmos para diferentes valores de relação sinal-ruído. Os resultados mostrados na Tabela IV indicam claramente a superioridade do algoritmo FQRD-RLS.

TABELA I

DESEMPENHOS DOS ALGORITMOS NLMS, BNDR-LMS E FQRD-RLS EM TERMOS DE SNR_{dB} APÓS PROCESSAMENTO PARA SINAIS DE VOZ CORROMPIDOS COM DIFERENTES VALORES DE SNR_{dB} DE ENTRADA (*in*).

SNR_{dB} (*in*)	NLMS	BNDR-LMS	FQRD-RLS
-10	-3.2718	-1.0149	4.7655
-5	1.0740	3.1553	7.5848
0	4.4838	6.2564	9.0682
5	6.5402	7.9965	9.6680

Levando-se em conta que o processamento é *off-line*, é possível melhorar razoavelmente os resultados já apresentados até o momento, principalmente no que diz respeito ao tempo de convergência inicial: embora não feito para a obtenção da tabela anterior, podemos rodar o filtro adaptativo uma vez para obter os coeficientes após a convergência (no final do sinal de voz, por exemplo) e rodar uma segunda vez inicializando o vetor de coeficientes com estes valores obtidos da primeira rodada. Tal procedimento será tão mais eficiente quanto mais estacionário for o ambiente onde é realizada a gravação. Se este ambiente não mudasse em nada suas estatísticas, poderíamos tirar uma média dos últimos coeficientes após a convergência e filtrar o sinal corrompido com um filtro fixo, resultado desta média. De qualquer maneira, mesmo com um sistema não estacionário, é muito provável que inicializar com o vetor de coeficientes do final seja melhor que inicializar com zeros (como normalmente é feito). Quanto aos algoritmos FQRD-RLS que não possuem os coeficientes disponíveis em cada iteração (como é o caso dos algoritmos NLMS e BNDR-LMS), simplesmente inicializamos as variáveis internas com os valores obtidos ao final da primeira rodada.

A motivação para este trabalho veio de um caso real trabalhado por um co-autor e cuja gravação não logrou um resultado favorável, ao que tudo indica, pela introdução de uma não-linearidade provocada pelo volume alto do sinal de áudio interferidor (áudio de uma novela). Um algoritmo adaptativo não linear está sendo pesquisado para resolver este problema.

Para aplicações reais, por não termos o padrão (sinal limpo) para podemos efetuar uma medida objetiva (tal como nos experimentos relatados), o resultado é meramente subjetivo. De fato, das demonstrações feitas na apresentação deste artigo, percebe-se claramente uma melhoria (inclusive percebe-se a convergência do filtro adaptativo).

No caso dos resultados obtidos não foram tão bons quanto os obtidos com as gravações realizadas pelos autores, as principais causas possíveis são:

1) não-linearidade (foi simulado tal efeito e o filtro adaptativo linear não elimina o ruído neste caso), por exemplo, do equipamento de som com volume alto;
2) ordem do filtro insuficiente (pode ocorrer, por exemplo, quando o ambiente tem dimensões maiores que o previsto neste artigo);
3) baixa qualidade do sinal gravado (microfone de baixa qualidade, escondido debaixo de um casaco e roçando no mesmo o tempo todo) ou sinal com muito ruído.

V. CONCLUSÃO

Este artigo visa mostrar as possibilidade de aplicação de um algoritmo adaptativo no auxílio das tarefas de um perito criminal; ele busca explicar, com uma linguagem simplificada, os principais parâmetros que regulam o desempenho desta técnica.

Percebe-se que o uso de algoritmos adaptativos da família FQRD-RLS são os que oferecem os melhores resultados mas necessitam de uma implementação numa linguagem de alto nível para que sejam viáveis para aplicações em casos reais.

Em aplicações comerciais (softwares comerciais usados em eliminação de ruído), é muito comum o uso de algoritmos NLMS no domínio da frequência. Esta possibilidade foi aplicada ao problema em questão e, embora não comentada anteriormente, não obteve resultado melhor que o proposto neste trabalho.

Por fim, chegamos a uma versão básica do programa aqui denominado *IMECleanSpeech* rodando parte em Matlab (sincronismo e estimação dos parâmetros) e parte em uma função executável (filtro adaptativo baseado no algoritmo FQRD-RLS). Espera-se, para breve, uma implementação final do programa, com todas suas funcionalidades em C e com uma versão executável que seja amigável a um usuário sem profundos conhecimentos em processamento digital de sinais.

REFERÊNCIAS

[1] P. S. R. Diniz, Adaptive Filtering: Algorithms and Practical Implementation. 3rd edition Springer, New York, NY, USA (2008).

[2] Stephen McGovern, Room Impulse Response Generator, versão 3.2, disponível em http://www.mathworks.com/matlabcentral/fileexchange.

[3] J. A. Apolinário Jr., C. A. Medina S., and P. S. R. Diniz, Infinite precision analysis of the fast QR algorithms based on backward prediction errors, Revista da Sociedade Brasileira de Telecomunicações (December 2002).

[4] J. A. Apolinário Jr., M. G. Siqueira, and P. S. R. Diniz, Fast QR algorithms based on backward prediction errors: a new implementation and its finite precision performance, Birkhäuser Circuits, Systems, and Signal Processing (July/August 2003).