Considerações sobre o Desempenho e a Complexidade Computacional de Algoritmos Utilizados em Sistemas de Cancelamento Adaptativo de Ruído

Rafael Rodrigo Pertum e Eduardo Vinicius Kuhn

***Resumo*—Este artigo trata da avaliação de desempenho e da complexidade computacional de um sistema de cancelamento adaptativo de ruído (implementado em hardware), operando com o algoritmo NLMS (*normalized leaste-mean-square*) e outros três algoritmos VSS-NLMS (*variable-step-size* NLMS) da literatura. Especificamente, sinais de fala contaminados por ruído de balbúrdia e de obras em construção são processados pelo sistema, sendo os sinais resultantes utilizados para avaliar o desempenho do sistema frente a diferentes valores de SNR (*signal-to-noise ratio*). Resultados obtidos por meio de avaliações objetivas de qualidade e inteligibilidade confirmam a eficácia do sistema. Ainda, com base no tempo de execução de cada função, uma comparação da complexidade computacional dos algoritmos é apresentada, evidenciando assim requisitos computacionais e características importantes dos algoritmos considerados.**

***Palavras-chave*—Algoritmo NLMS, algoritmos VSS-NLMS, cancelamento adaptativo de ruído, filtragem adaptativa.**

***Abstract*—This paper deals with performance evaluation and computational complexity analysis of an adaptive noise canceling system (implemented in hardware), operating with the normalized least-mean-square (NLMS) algorithm and other three variable step-size NLMS (VSS-NLMS) algorithms from the literature. Specifically, speech signals contaminated by babble noise and construction noise are processed through the system, and the results are used to assess the performance of the system under different values of signal-to-noise ratio (SNR). Results obtained by objective (quality and intelligibility) evaluation confirm the effectiveness of the system. Still, based on the execution time of each function used, a comparison between the computational complexity of the algorithms is presented, thus evidencing computational requirements and important characteristics of the algorithms considered.**

***Keywords*—NLMS algorithm, VSS-NLMS algorithms, adaptive noise canceling, adaptive filtering.**

I. INTRODUÇÃO

Atualmente, sistemas de cancelamento adaptativo de ruído (ANC, *adaptive noise cancelling*) vêm sendo utilizados em diversas aplicações práticas, tais como na supressão de ruído em aparelhos auditivos, celulares e/ou de áudio-conferência [1]–[3]. Nessas aplicações, busca-se mitigar o ruído que degrada o sinal de interesse, no intuito de facilitar a interpretação da informação desejada e/ou melhorar a qualidade da comunicação. Para tal, técnicas de filtragem adaptativa são empregadas para sintetizar (em tempo real) uma estimativa do ruído que corrompe o sinal de interesse para, então, atenuá-lo (por subtração) no domínio elétrico [1]. Dentre os algoritmos adaptativos comumente utilizados em tais aplicações, se destacam o algoritmo NLMS (*normalized least-mean-square*) e alguns importantes algoritmos VSS-NLMS (*variable step-size* NLMS) da literatura [1], [4]–[6].

Na prática, a implementação de um sistema de ANC está sujeita a outros fatores de projeto que, geralmente, não são tratados em simulações conduzidas para avaliar o desempenho de algoritmos adaptativos. Como exemplo desses fatores de projeto, é possível citar a capacidade de processamento do microcontrolador, os requisitos de memória, a taxa de amostragem utilizada, as características dos sinais envolvidos, os cenários de operação, ou ainda o número de bits do conversor A/D (analógico/digital). Negligenciar tais fatores pode prejudicar o desempenho de um sistema de ANC como um todo e, consequentemente, afetar a qualidade final do sinal processado [1]. Portanto, torna-se imperativo analisar o tempo de execução e a complexidade computacional dos algoritmos implementados, assim como empregar metodologias de avaliação subjetivas [7] e/ou objetivas [8], [9] para verificar a qualidade do sinal de fala obtido como resultado de algum processamento em condições de operação mais realistas.

Neste contexto, o presente trabalho de pesquisa tem os seguintes objetivos:

i) revisitar a implementação do sistema de ANC de [10], a placa de desenvolvimento utilizada [11] e os algoritmos adaptativos considerados [1], [4]–[6];
ii) investigar (visualmente) os espectrogramas do sinal original, do sinal corrompido por ruído e do sinal processado pelo sistema de ANC;
iii) avaliar o desempenho dos algoritmos considerados usando métricas objetivas de qualidade e inteligibilidade frente a condições mais realistas de operação; e
iv) determinar a complexidade computacional dos algoritmos em termos do tempo de execução das funções utilizadas e do tempo requerido para processar um quadro do sinal.

Note que o presente trabalho estende os resultados alcançados em [10], versando agora sobre o desempenho do sistema operando em cenários de ruído mais realistas (para além do caso de ruído branco gaussiano) e sobre a complexidade computacional dos algoritmos considerados.

II. FUNDAMENTAÇÃO TEÓRICA

Nesta seção, o sistema de ANC considerado é descrito e as expressões do algoritmo NLMS [1] bem como daqueles introduzidos em [4]–[6] são brevemente revisitadas.

A. Topologia de um sistema de ANC

Em um sistema de ANC, um algoritmo adaptativo [empregando um sinal de referência $x(n)$ e o sinal de erro $e(n)$ no instante de tempo n] ajusta os coeficientes de uma estrutura de filtragem de forma a produzir uma estimativa do ruído $d(n)$ que corrompe o sinal de interesse $s(n)$ presente na entrada primária. Então, realizando a subtração entre o sinal corrompido por ruído e a estimativa do ruído $\hat{d}(n)$, obtém-se o sinal de erro como

$$e(n) = s(n) + d(n) - \hat{d}(n). \quad (1)$$

Dessa forma, conforme o algoritmo converge para a solução em regime permanente [isto é, $\hat{d}(n) \to d(n)$ para $n \to \infty$], o sinal de erro tende para o sinal de interesse [isto é, $e(n) \to s(n)$ para $n \to \infty$], produzindo assim a atenuação do ruído na saída do sistema (como desejado).

Rafael Rodrigo Pertum e Eduardo Vinicius Kuhn estão vinculados ao LAPSE–Laboratório de Processamento de Sinais e Eletrônica do Departamento de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná (UTFPR), Toledo, PR, Brasil (e-mails: pertum@alunos.utfpr.edu.br e kuhn@utfpr.edu.br).

B. Algoritmo NLMS

Considerando uma estrutura transversal de filtragem, a equação de adaptação dos coeficientes do algoritmo NLMS pode ser definida como [1]

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu \frac{\mathbf{x}(n)e(n)}{\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n) + \varepsilon} \tag{2}$$

na qual $\mathbf{w}(n) = [w_0(n) \ \cdots \ w_{M-1}(n)]^{\mathrm{T}}$ denota o vetor de coeficientes do filtro adaptativo de ordem M, $(\cdot)^{\mathrm{T}}$, o operador de transposição, $\mathbf{x}(n) = [x(n) \ \cdots \ x(n-M+1)]^{\mathrm{T}}$, um vetor contendo as amostras mais recentes da entrada de referência, $0 < \mu < 2$, o passo de adaptação e $\varepsilon > 0$, um parâmetro de regularização usado para prevenir divisão por zero e estabilizar a solução.

C. Algoritmo VSS-NLMS de Shin

Em algoritmos de passo variável, o passo de adaptação em (2) é feito variante no tempo, isto é, μ é substituído por $\mu(n)$. Dessa forma, (2) pode ser revisada para [1]

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(n) \frac{\mathbf{x}(n)e(n)}{\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n) + \varepsilon} \tag{3}$$

sendo o valor do passo de adaptação $\mu(n)$ ajustado iterativamente de acordo com alguma regra específica. Particularmente, no algoritmo VSS-NLMS de Shin [4], o ajuste do passo de adaptação é realizado de acordo com

$$\mu(n) = \mu_{\text{máx}} \frac{\mathbf{p}^{\mathrm{T}}(n)\mathbf{p}(n)}{\mathbf{p}^{\mathrm{T}}(n)\mathbf{p}(n) + C} \tag{4}$$

onde

$$\mathbf{p}(n) = \beta \mathbf{p}(n-1) + (1-\beta) \frac{\mathbf{x}(n)e(n)}{\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n)} \tag{5}$$

com $0 < C \ll 1$ denotando uma constante positiva inversamente proporcional a SNR (*signal-to-noise ratio*), $\mu_{\text{máx}}$, o valor máximo que o passo de adaptação pode assumir e $0 \ll \beta < 1$, uma constante positiva de suavização.

D. Algoritmo VSS-NLMS de Benesty

No algoritmo VSS-NLMS de Benesty (dito não paramétrico) [5], o passo de adaptação é computado por

$$\mu(n) = \begin{cases} \gamma(n), & \text{se } \hat{\sigma}_e(n) \geq \hat{\sigma}_v \\ 0, & \text{caso contrário} \end{cases} \tag{6}$$

com

$$\gamma(n) = 1 - \frac{\hat{\sigma}_v}{\delta + \hat{\sigma}_e(n)} \tag{7}$$

onde δ representa uma constante de regularização, $\hat{\sigma}_e^2(n)$ denota uma estimativa da variância do sinal de erro obtida de

$$\hat{\sigma}_e^2(n) = \lambda_1 \hat{\sigma}_e^2(n-1) + (1-\lambda_1) e^2(n) \tag{8}$$

para

$$\lambda_1 = 1 - \frac{1}{KM}, \ K \geq 2 \tag{9}$$

e $\hat{\sigma}_v^2$ caracteriza uma estimativa da variância do ruído de medição. Vale salientar que Ciochina, Paleologu e Benesty [12] sugerem que essa estimativa seja obtida de acordo com [13], isto é,

$$\hat{\sigma}_v^2 = \hat{\sigma}_e^2(n) - \frac{1}{\hat{\sigma}_x^2(n)} \hat{\mathbf{r}}_{ex}^{\mathrm{T}}(n) \hat{\mathbf{r}}_{ex}(n) \tag{10}$$

com

$$\hat{\sigma}_x^2(n) = \lambda_2 \hat{\sigma}_x^2(n-1) + (1-\lambda_2) x^2(n) \tag{11}$$

e

$$\hat{\mathbf{r}}_{ex}(n) = \lambda_2 \hat{\mathbf{r}}_{ex}(n-1) + (1-\lambda_2)\mathbf{x}(n)e(n). \tag{12}$$

Note que λ_1 e λ_2 representam constantes de suavização distintas e, portanto, podem assumir valores diferentes.

E. Algoritmo VSS-NLMS de Zipf

Para o algoritmo VSS-NLMS de Zipf [6], a regra de ajuste para o passo de adaptação em (3) é descrita por meio de

$$\mu(n) = \frac{p^2(n)}{q^2(n)} \tag{13}$$

com

$$p(n) = \beta p(n-1) + (1-\beta) e(n) e(n-1) \tag{14}$$

denotando uma estimativa de correlação do sinal de erro e

$$q(n) = \beta q(n-1) + (1-\beta) e^2(n) \tag{15}$$

representando uma estimativa da potência do sinal de erro, nas quais $0 \ll \beta < 1$ define um parâmetro de suavização (detalhes sobre o funcionamento do algoritmo são discutidos em [6]).

III. Revisitando a Implementação Proposta

Na implementação do sistema de ANC, foi utilizada uma placa de desenvolvimento Cortex-FM4 *Starter* da *Cypress Semiconductors*, a qual possui um microcontrolador ARM® Cortex-M4 S6E2CC operando a 200 MHz [11]. Essa placa de desenvolvimento conta com um *codec* (*coder-decoder*) Wolfson® WM8731 responsável por realizar a interface entre entrada/saída de áudio (através de conectores do tipo P2 estéreo) e o microcontrolador por meio do protocolo de comunicação I2S (*inter-IC sound*). Salienta-se que os sinais de áudio (fala) envolvidos foram digitalizados com uma frequência de amostragem de 8 kHz, cada amostra quantizada em 16 bits e o processamento realizado em quadros (*frames*) de 128 amostras. Por sua vez, o desenvolvimento dos códigos foi realizado utilizando a IDE (*integrated development environment*) Keil µVision® MDK (*microcontroller development kit*) versão 5 fornecida pela ARM® [14].

Tabela 1. Valores utilizados para os parâmetros dos algoritmos considerados.

NLMS	VSS-NLMS de Shin	VSS-NLMS de Benesty	VSS-NLMS de Zipf
$\mu = 0,05$	$C = 0,001$	$\lambda_1 = 0,97$	$\beta = 0,99$
$\varepsilon = 1,0$	$\beta = 0,9961$	$\lambda_2 = 0,9987$	$\varepsilon = 1,0$
	$\mu_{\text{máx}} = 1,5$	$\delta = 1,0$	
	$\varepsilon = 1,0$	$\varepsilon = 20\hat{\sigma}_x^2$	

```
//produto interno entre w[] e x[]
arm_dot_prod_f32(x,w,M,&d_chapeu);
//determinação do erro
e=d-d_chapeu;
//estimativa da variância do sinal de erro e de x[]
var_e=lambda1*(var_e)+(1-lambda1)*e*e;
var_x=lambda2*(var_x)+(1-lambda2)*x[0]*x[0];
//define o parâmetro épsilon
ep=20*var_x;
//correlação entre x[] e o erro (r_ex = A + B)
arm_scale_f32(r_ex,lambda2,A,M);
arm_scale_f32(x,(1-lambda2)*e,B,M);
arm_add_f32(A,B,r_ex,M);
arm_power_f32(r_ex,M,&energia_r_ex);
//variância do sinal de fala (ruído de medição)
var_v=var_e-(energia_r_ex/var_x);
//cálculo do desvio padrão (erro e ruído de medição)
arm_sqrt_f32(var_e,&desv_e);
arm_sqrt_f32(var_v,&desv_v);
//fator gama
gama=(1-desv_v/(delta+desv_e));
//atualiza os coeficientes do filtro
if(desv_e>=desv_v){
      mu=gama;
      arm_scale_f32(x,mu/(ep+energia_x)*e,fator,M);
      arm_add_f32(w,fator,w,M);}
```

Fig. 1. Trecho de código da implementação (revisada) do algoritmo VSS-NLMS de Benesty.

Levando em conta a linguagem de programação C, trechos de código relacionados à implementação de cada um dos quatro algoritmos adaptativos considerados aqui são fornecidos em [10, Figs. 2-5]. Especificamente, a implementação do algoritmo NLMS (descrito na Seção II-B) é mostrada em [10, Fig. 2], do algoritmo VSS-NLMS de Shin (revisitado na Seção II-C) em [10, Fig. 3], do algoritmo VSS-NLMS de Benesty (discutido na Seção II-D) é aqui revisada conforme consta na Fig. 1, enquanto do algoritmo VSS-NLMS de Zipf (dado na Seção II-E) segue como em [10, Fig. 5]. Em tais implementações, foram utilizadas funções de processamento de sinais fornecidas na biblioteca CMSIS-DSP (*Cortex microcontroller software interface standard–digital signal processing*) versão 1.7.1, objetivando otimizar o código e facilitar a implementação. Vale destacar que a estrutura de filtragem tem $M = 128$ coeficientes e que os valores dos parâmetros dos diferentes algoritmos foram ajustados conforme indicado na Tabela 1.

IV. Resultados e Discussão

Tendo em vista os objetivos estabelecidos no presente trabalho, esta seção apresenta e discute os resultados experimentais obtidos na avaliação do sistema de ANC considerado. Primeiramente, uma análise visual do espectrograma dos sinais envolvidos é apresentada com a finalidade de verificar e validar o sistema (veja a Seção IV-A). Em seguida, comparações com respeito à qualidade e inteligibilidade dos sinais de fala originais e processados são realizadas considerando três metodologias da literatura, a saber: i) *perceptual evaluation of speech quality* (PESQ) [8], ii) *short-time objective intelligibility measure* (STOI) [9] e iii) *application programming interface* (API) *speech-to-text* da Google Inc. [15] (veja as Seções IV-B e IV-C). Por último, é determinada a complexidade computacional dos algoritmos em termos do tempo de execução das principais funções utilizadas e do tempo requerido para processar um quadro de 128 amostras de sinal; dessa forma, torna-se possível compreender melhor os requisitos computacionais dos algoritmos considerados (veja a Seção IV-D).

Para conduzir os experimentos, sinais de áudio foram sintetizados com o auxílio de *scripts*[1] em linguagem Python a fim de emular aqueles observados na entrada primária e de referência do sistema de ANC. Especificamente, são consideradas dezesseis sentenças em português do Brasil [16] para compor os sinais de fala e três gravações da base *Noise Recordings* [17, Appendix C] para compor os cenários de ruído. Desses cenários, o ruído de balbúrdia é obtido de *cafeteria_babble.wav* e o de obras em construção, a partir de *Construction_Crane_Moving.wav* e *Construction_Jackhammer2.wav*. Esses sinais de ruído são ajustados em intensidade, filtrados por uma planta obtida de [18] e adicionados aos sinais de fala utilizando a rotina "*addnoise_asl.m*" (dada em [17, Appendix C]). Dessa forma, 544 sinais (entrada primária) são produzidos, compreendendo uma faixa de SNR de -12 dB a 20 dB. Tais sinais servem de entrada para o sistema implementado na placa de desenvolvimento, sendo o canal esquerdo da entrada *line-in* destinado à entrada de referência (ruído) e o direito à entrada primária (sinal ruidoso). Por fim, é então realizada a aquisição do sinal de erro observado na saída do sistema para cada um dos sinais de entrada, produzindo assim os resultados experimentais (como ilustrado na Fig. 2).

[1] Os scripts utilizados e os resultados obtidos estão disponíveis em [21].

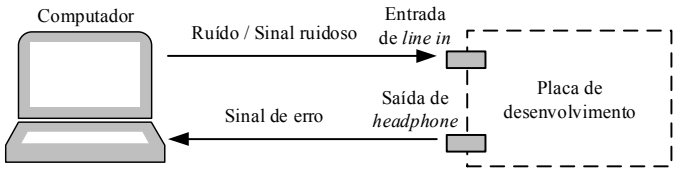

Fig. 2. Diagrama de blocos da estratégia de avaliação considerada.

A. Análise dos sinais envolvidos via espectrograma

Na Fig. 3, são apresentados os espectrogramas de um dos sinais utilizados aqui para realizar as avaliações. Especificamente, a Fig 3(a) ilustra o espectrograma do sinal de fala original (limpo), a Fig. 3(b) traz o espectrograma do sinal contaminado por ruído de balbúrdia com 0 dB de SNR, enquanto a Fig. 3(c) ilustra o espectrograma do sinal obtido como resultado do processamento pelo sistema de ANC. Ao comparar as Figs. 3(a) e 3(b), observa-se o efeito da adição do ruído de balbúrdia no espectro sinal de fala (resultando no sinal contaminado), uma vez que as características espectrais do sinal original foram perdidas. Já a partir da Fig. 3(c), verifica-se a boa capacidade do sistema de ANC implementado para remover o ruído presente no sinal de fala contaminado [Fig. 3(b)], visto que as características espectrais do sinal original [Fig. 3(a)] foram recuperadas. Note ainda que, comparando as Figs. 3(a) e 3(c), existem características espectrais do ruído presente nos primeiros segundos do sinal processado, o que se deve ao tempo de convergência do algoritmo adaptativo.

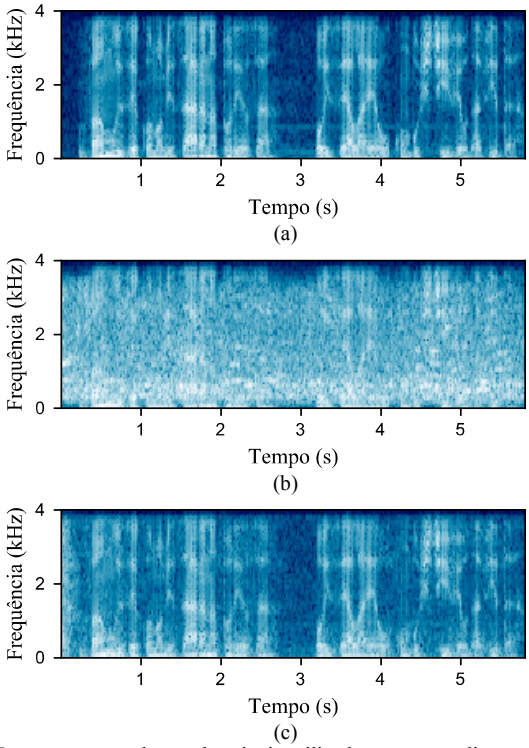

Fig. 3. Espectrogramas de um dos sinais utilizados para a avaliação. (a) Sinal de fala original (limpo). (b) Sinal de fala contaminado com ruído de balbúrdia com 0 dB de SNR. (c) Sinal de fala após o processamento pelo sistema de ANC operando com o algoritmo VSS-NLMS de Benesty.

B. Avaliação de qualidade

A Recomendação P.862.1 da *International Telecomunication Union-Telecommunication Standardization Sector* (ITU-T) estabelece a PESQ *mean opinion score-listening quality objective* (MOS-LQO) como um método de avaliação objetiva de qualidade de voz [8]. A PESQ (MOS-LQO) consiste em comparar um áudio de referência (sinal original) com um áudio resultante de algum tipo de processamento em meio digital, o qual pode ter sofrido

degradação. Essa avaliação é realizada via processamento computacional e tem como resultado uma pontuação entre 1 e 4,5 (quanto menor a pontuação, mais baixa a qualidade do áudio avaliado). Então, dispondo da implementação da PESQ fornecida em [8, ITU-T P.862.1], as pontuações obtidas para cada sinal foram computadas. A partir dessas pontuações, a média dentre os diferentes sinais foi contabilizada para cada valor de SNR e os resultados obtidos são ilustrados na Fig. 4(a), para o cenário de ruído de balbúrdia, e na Fig. 4(b), para o cenário de ruído de obras em construção. Observa-se em tais figuras que o algoritmo VSS-NLMS de Benesty apresentou o melhor resultado "global" para ambos os cenários de ruído, seguido dos algoritmos NLMS, VSS-NLMS de Shin e de Zipf. Destaca-se também que, em certos intervalos de SNR, todos os algoritmos apresentaram resultados de qualidade superiores quando comparados com o sistema de ANC desligado; por exemplo, o algoritmo VSS-NLMS de Zipf entre -12 dB e 6 dB, o algoritmo VSS-NLMS de Shin entre -12 dB e 12 dB, e os algoritmos NLMS e VSS-NLMS de Benesty na faixa de -12 dB até 16 dB. Portanto, sob certas condições, é possível inferir que os algoritmos considerados propiciaram uma melhoria importante de qualidade no sinal de fala processado se comparado ao do sistema de ANC desligado.

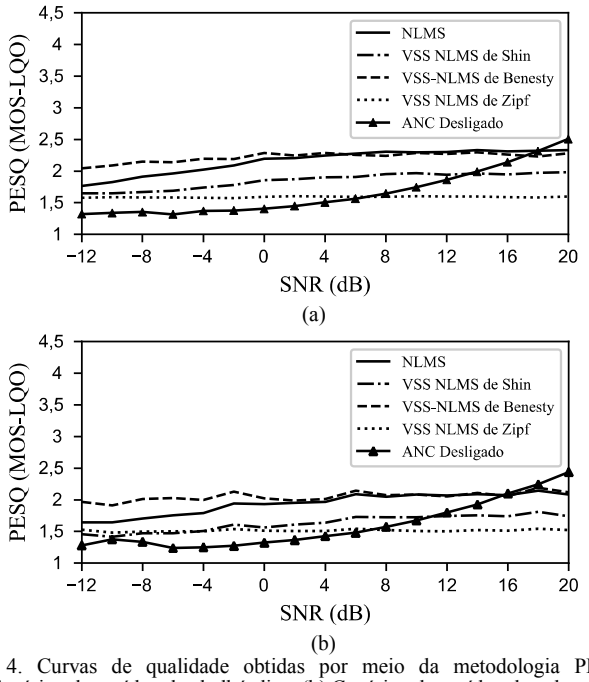

Fig. 4. Curvas de qualidade obtidas por meio da metodologia PESQ. (a) Cenário de ruído de balbúrdia. (b) Cenário de ruído de obras em construção.

C. Avaliação de inteligibilidade

A STOI e a API *speech-to-text* da Google Inc. foram utilizadas para conduzir avaliações objetivas de inteligibilidade. Empregando um áudio de referência (assim como na PESQ), a STOI é utilizada para mensurar quanto a inteligibilidade de um sinal obtido como resultado de algum processamento foi degradada, resultando em uma pontuação de 0 a 1 (0 indica pior inteligibilidade e 1, a melhor). A partir da implementação da metodologia STOI disponível em [19], resultados experimentais foram computados e o desempenho médio obtido para cada valor de SNR é fornecido na Fig. 5. Adicionalmente, a API *speech-to-text* da Google Inc. (disponível em [20]) é utilizada aqui para contabilizar a taxa de acerto do número de palavras corretamente reconhecidas pelo conversor fala-texto. Para isso, sinais de áudio resultantes do processamento são submetidos à API *speech-to-text* a fim de se obter uma transcrição do sinal, comparada então com a transcrição fornecida em [16]. Em seguida, a taxa de acerto é computada como a razão entre o número total de palavras reconhecidas corretamente e o número total de palavras presentes nas transcrições das sentenças de referência. Vale destacar que as palavras corretamente reconhecidas são aquelas que possuem mais de três letras e pertencem a sentença de referência analisada. Dessa forma, os resultados experimentais foram computados para cada valor de SNR e o desempenho em termos de taxa de acerto é apresentado na Fig. 6. Nessas avaliações de inteligibilidade, os algoritmos VSS-NLMS de Benesty e NLMS apresentaram os melhores resultados tanto no cenário de ruído de balbúrdia [Figs. 5(a) e 6(a)] quanto no de obras em construção [Figs. 5(b) e 6(b)]. Ainda para valores de SNR de até 8 dB, todos os algoritmos (exceto aquele proposto por Zipf) apresentaram um aumento de inteligibilidade quando comparados com o sistema de ANC desligado. O pior resultado foi observado para o algoritmo VSS-NLMS de Zipf; apesar disso, esse algoritmo ainda é superior ao resultado obtido com o sistema de ANC desligado para valores de SNR abaixo de 2 dB (de acordo com a Fig. 5) e -2 dB (conforme mostra a Fig. 6). Por último, uma degradação de desempenho do sistema é observada frente a condições de ruído mais realistas, quando os resultados alcançados aqui são comparados com aqueles de [10] para ruído branco gaussiano.

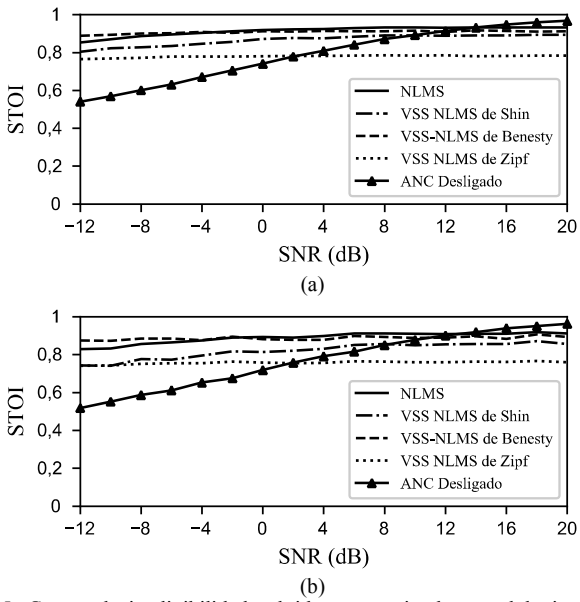

Fig. 5. Curvas de inteligibilidade obtidas por meio da metodologia STOI. (a) Cenário de ruído de balbúrdia. (b) Cenário de ruído de obras em construção.

D. Complexidade computacional

Para avaliar a complexidade computacional, foi utilizado um contador de 32-/16-bit (isto é, um *dual timer*) disponível no microcontrolador, o qual realiza uma contagem decrescente com base em um temporizador interno. Com esse contador, foi contabilizado o tempo para processar um quadro de 128 amostras (veja a Tabela 2) e o tempo para executar as principais funções utilizadas (veja a Tabela 3). Na Tabela 2, é expresso o tempo de processamento (em ms) para cada algoritmo considerado, a ocupação (percentual) da capacidade de processamento do microcontrolador e o tempo de execução normalizado em função do algoritmo NLMS. Este último, representa quantas vezes cada implementação é mais complexa (em termos de tempo de execução) quando comparada com aquela do algoritmo NLMS. Esses resultados mostram que os algoritmos NLMS e VSS-NLMS de Zipf

possuem o menor tempo de execução (2,77 ms e 2,89 ms, respectivamente), ocupando assim apenas 18% da capacidade de processamento do microcontrolador. Em contrapartida, os algoritmos VSS-NLMS de Shin e de Benesty exibem os maiores tempos de execução (5,12 ms e 5,31 ms, respectivamente) e, consequentemente, maior ocupação do microcontrolador (em torno de 33%); portanto, são aproximadamente 2 vezes mais complexos do que o algoritmo NLMS. Já na Tabela 3, apresenta-se o tempo de execução para cada função utilizada em µs, o número de vezes que tais funções são empregadas (em cada algoritmo) e o tempo total acumulado para processar um quadro de 128 amostras. Dentre elas, as funções `arm_add_f32()` e `arm_dot_prod_f32()` demandaram o maior tempo de execução, as quais desempenham a soma entre dois vetores (5,32 µs) e o produto interno entre dois vetores (5,15 µs). Observa-se ainda que o tempo total acumulado das funções utilizadas (Tabela 3) e o tempo de processamento de cada algoritmo (Tabela 2) é muito próximo, ratificando os resultados obtidos. A diferença entre esses valores pode ser atribuída à outras operações, tais como subtração, divisão e atribuição necessárias para a operação dos algoritmos. Portanto, a partir da Tabela 3, é possível ainda obter uma estimativa do tempo de execução de um dado algoritmo arbitrário levando em conta o tempo de execução acumulado das principais funções utilizadas em sua correspondente implementação, sendo esse um importante critério de projeto.

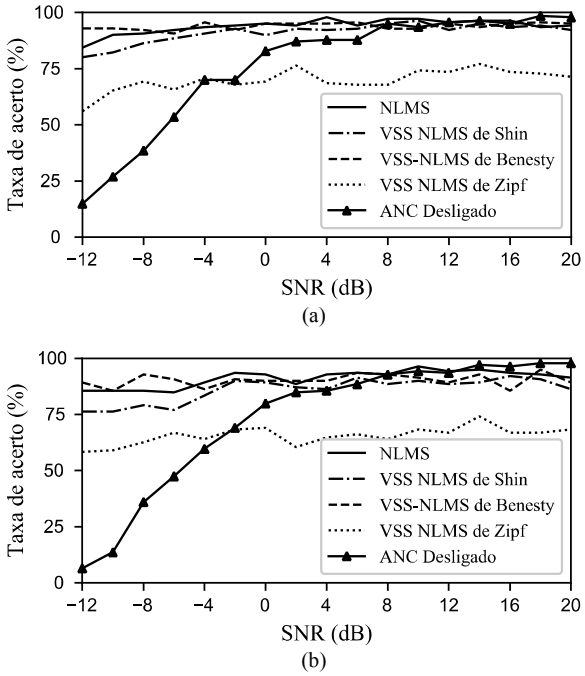

Fig. 6. Curvas de inteligibilidade obtidas a partir da taxa de acerto no reconhecimento de palavras via API *speech-to-text* do Google Inc. (a) Cenário de ruído de balbúrdia. (b) Cenário de ruído de obras em construção.

V. Considerações Finais

Neste artigo, considerando o algoritmo NLMS e três outros algoritmos VSS-NLMS da literatura, avaliações de desempenho e de complexidade computacional de um sistema de ANC foram apresentadas. Tais avaliações contemplam a análise visual dos sinais envolvidos via espectrogramas, metodologias objetivas de qualidade e inteligibilidade e, também, o tempo de execução dos códigos pertinentes às implementações propostas. A partir dos resultados alcançados, foi possível confirmar a eficácia do sistema de ANC em certos intervalos de SNR, bem como identificar alguns requisitos computacionais entre os algoritmos considerados. Futuros trabalhos de pesquisa poderiam focar na implementação de outros algoritmos da literatura, na construção de um arranjo de microfones para integrar ao sistema, ou ainda na utilização da metodologia *Perceptual Objective Listening Quality Analysis* (POLQA) para determinar a qualidade dos sinais processados.

Tabela 2. Tempo de processamento de um quadro de 128 amostras.

Algoritmo	Tempo (ms)	% Ocupação	/ NLMS
NLMS	2,77	17,29	1
VSS-NLMS de Shin	5,12	32,00	1,85
VSS-NLMS de Zipf	2,89	18,06	1,04
VSS-NLMS de Benesty	5,31	33,14	1,92

Tabela 3. Tempo de execução acumulado das principais funções utilizadas por cada algoritmo para processar um quadro de 128 amostras.

Função	Tempo (µs)	Algoritmo			
		NLMS	Shin	Benesty	Zipf
`arm_power_f32()`	3,54	×1	×2	×1	×1
`arm_dot_prod_f32()`	5,15	×1	×1	×1	×1
`arm_scale_f32()`	4,55	×1	×3	×3	×1
`arm_add_f32()`	5,32	×1	×2	×2	×1
`memcpy()`	2,08	×1	×1	×1	×1
Total acumulado (ms)		2,64	4,94	4,49	2,64

Referências

[1] S. Haykin, *Adaptive Filter Theory*, 5th ed. Upper Saddle River, NJ: Prentice-Hall, 2014.

[2] I. Panahi, N. Kehtarnavaz, and Thibodeau, "Smartphone-based noise adaptive speech enhancement for hearing aid applications", in *Proc. IEEE Int. Conf. Eng. Med. Biol. Soc. (EMBC)*, Orlando, FL, USA, Aug. 2016, pp. 85-88.

[3] A. Sugiyama, R. Miyahara, and K. Oosugi, "A noise robust hearable device with an adaptive noise canceller and its DSP implementation" in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Brighton, UK, May 2019, pp. 2722-2726.

[4] H.-C. Shin, A. H. Sayed, and W.-J. Song, "Variable step-size NLMS and affine projection algorithms", in *IEEE Signal Process. Letters*, vol. 11, pp. 132-135, Feb. 2004.

[5] J. Benesty, H. Rey, L. R. Vega, and S. Tressens, "A nonparametric VSS NLMS algorithm", in *IEEE Signal Process. Letters*, vol. 13, pp. 581-584, Oct. 2006.

[6] J. G. F. Zipf, O. J. Tobias, and R. Seara, "Non-parametric VSS-NLMS algorithm with control parameter based on the error correlation," in *Proc. IEEE Int. Telecommun. Symp.*, Manaus, AM, Brazil, Sep. 2010, pp. 1-5.

[7] ITU-T Recommendation P.800 - *Methods for subjective determination of transmission quality*, Geneva, Switzerland: Int. Telecomm. Union, 1996.

[8] ITU-T Recommendation P.862 - *Perceptual evaluation of speech quality (PESQ)*, Geneva, Switzerland: Int. Telecomm. Union, 2005.

[9] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "An algorithm for intelligibility prediction of time-frequency weighted noisy speech", *IEEE Trans. Audio Speech Lang. Process.*, vol. 19, pp. 2125-2136, Sep. 2011.

[10] R. R. Pertum e E. V. Kuhn, "Implementação e Avaliação de Desempenho de Algoritmos para Cancelamento Adaptativo de Ruído", in *Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT)*, Petrópolis, RJ, Brasil, Out. 2019.

[11] Cypress Semiconductors, *ARM Cortex-FM4 Starter kit (S6E2CC-ETH)*, Disponível em: http://www.cypress.com/documentation/development-kitsboards/sk-fm4-176l-s6e2cc-fm4-family-quick-start-guide

[12] S. Ciochina, C. Paleologu, and J. Benesty, "An optimized NLMS algorithm for system identification," *Signal Process.*, vol. 118, pp. 115-121, Jan. 2016.

[13] M. A. Iqbal and S. L. Grant, "Novel variable step size NLMS algorithms for echo cancellation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP),* Las Vegas, NV, 31 Mar.-4 Apr. 2008, pp. 241-244.

[14] ARM, *Keil µVision® MDK*, Disponível em: https://www.keil.com

[15] Google Inc. (20 de abril 2020). *Cloud speech-to-text* [Online]. Disponível em: https://cloud.google.com/speech-to-text/

[16] ITU-T Recommendation P.50 - *Appendix I: Real speech recordings (Brazilian Portuguese)*, Geneva, Switzerland: Int. Telecomm. Union, 2000.

[17] P. C. Loizou, *Speech Enhancement: Theory and Practice*, 2nd ed. Boca Raton: CRC Press, 2017.

[18] ITU-T Recommendation G.168 - *Digital Network Echo Cancellers*. Geneva, Switzerland: Int. Telecomm. Union, Apr. 2015.

[19] M. Pariente, FR-J: Laboratoire des Systèmes Perceptifs. (14 de maio 2019). *pystoi*. [Online]. Disponível em: https://github.com/mpariente/pystoi

[20] A. Zhang, Hypotenuse Labs. (20 de abril 2020). *SpeechRecognition*. [Online]. Disponível em: https://github.com/Uberi/speech_recognition

[21] R. R. Pertum e E. V. Kuhn. (24 de abril 2020). *Scripts e resultados* [Online]. Disponível em: http://lapse.td.utfpr.edu.br/downloads/artigo_sbrt2020.zip