Proceedings of the VI Brazilian Conference on Neural Networks - VI Congresso Brasileiro de Redes Neurais
pp. 43–48, June 2–5, 2003 - Centro Universitário da FEI, São Paulo - SP - Brazil

Complexidade Computacional de um Algoritmo Competitivo Aplicado ao Projeto de Quantizadores Vetoriais

Francisco Madeiro[1], Waslon T. A. Lopes[2], Benedito G. Aguiar Neto[2], Marcelo S. Alencar[2]
[1]Universidade Católica de Pernambuco – Recife, PE, Brasil
[2]Universidade Federal de Campina Grande – Campina Grande, PB, Brasil*
madeiro@dei.unicap.br, {waslon,bganeto,malencar}@dee.ufcg.edu.br

Abstract

In the present paper, the computational complexity of a competitive learning algorithm applied to vector quantization (VQ) codebook design is investigated. Analytical expressions (as a function of the codebook size, the dimension of the codevectors, the number of training vectors and the number of iterations performed) are derived for the number of operations (multiplications, divisions, additions, subtractions and comparisons) performed by the competitive algorithm. Analytical expressions are also derived for the tradional LBG (Linde-Buzo-Gray) algorithm. Regarding VQ codebook design for image coding, results show that the computational complexity of the competitive algorithm is lower than that of LBG.

1. Introdução

O projeto de dicionários desempenha um papel importante para o bom desempenho de sistemas de processamento de sinais baseados em quantização vetorial (QV) [1, 2]. Em codificação de voz e imagem baseada em QV, a qualidade dos sinais reconstruídos depende dos dicionários projetados. Em sistemas de identificação de locutor que utilizam QV paramétrica, as taxas de identificação dependem dos dicionários de padrões acústicos de referência projetados para cada locutor cadastrado pelo sistema. Dentre as diversas técnicas para projeto de dicionários, o algoritmo LBG (Linde-Buzo-Gray) [3] destaca-se por sua ampla utilização. Outras abordagens têm sido utilizadas em projeto de dicionários, como por exemplo: algoritmo de Kohonen [4] e outros algoritmos não-supervisionados [5–9]; relaxação estocástica [10]; algoritmos *fuzzy* [11, 12] e algoritmo genético [13].

O presente trabalho tem como objetivo avaliar a complexidade computacional de um algoritmo competitivo aplicado ao projeto de dicionários para quantização vetorial. É apresentada uma avaliação comparativa desse algoritmo competivo e do algoritmo LBG no que diz respeito ao número de operações (multiplicações, divisões, adições, subtrações e comparações) envolvidas no projeto de dicionários.

*Os autores gostariam de expressar os agradecimentos à CAPES e ao CNPq pelo apoio financeiro ao trabalho.

O restante deste artigo encontra-se organizado de acordo com as seções a seguir. A Seção 2 apresenta uma abordagem suscinta da quantização vetorial. O algoritmo LBG é descrito na Seção 3. A Seção 4 apresenta uma breve descrição do algoritmo competitivo (AC). Na Seção 5, cujo foco é a complexidade computacional dos algoritmos LBG e AC, são obtidas expressões analíticas para o número de operações realizadas por esses algoritmos. Resultados e conclusões são apresentados, respectivamente, nas Seções 6 e 7.

2. Quantização Vetorial

A quantização vetorial [1, 2] pode ser definida como um mapeamento Q de um vetor de entrada $\boldsymbol{x}$ pertencente ao espaço euclidiano K-dimensional, $\mathbb{R}^K$, em um vetor pertencente a um subconjunto finito W de $\mathbb{R}^K$, ou seja,

$$Q : \mathbb{R}^K \rightarrow W. \tag{1}$$

O dicionário $W = \{\boldsymbol{w}_i;\ i = 1,\ 2,\ \ldots,\ N\}$ é o conjunto de vetores de reprodução (também denominados vetores-código ou vetores de reconstrução), K é a dimensão do quantizador vetorial e N é o tamanho do dicionário, isto é, o número de vetores-código (ou número de níveis, em analogia com a quantização escalar).

O mapeamento Q introduz um particionamento de $\mathbb{R}^K$ em N células (denominadas regiões de Voronoi) S_i, $i = 1,\ 2,\ \ldots,\ N$, tais que

$$\bigcup_{i=1}^{N} S_i = \mathbb{R}^K \text{ e } S_i \cap S_j = \emptyset \text{ para } i \neq j, \tag{2}$$

em que cada célula S_i é definida por

$$S_i = \{\boldsymbol{x} :\ Q(\boldsymbol{x}) = \boldsymbol{w}_i\}. \tag{3}$$

Mais precisamente,

$$S_i = \{\boldsymbol{x} :\ d(\boldsymbol{x}, \boldsymbol{w}_i) < d(\boldsymbol{x}, \boldsymbol{w}_j), \forall j \neq i\}, \tag{4}$$

em que $d(\cdot, \cdot)$ denota uma medida de distância. O vetor-código $\boldsymbol{w}_i$, portanto, constitui o vetor representativo de todos os vetores de entrada pertencentes à célula S_i.

A taxa de codificação de um quantizador vetorial, que mede o número de bits por componente de vetor, é dada por $R = \frac{1}{K}\log_2 N$. Em codificação de forma de onda de voz (*e. g.* [14]), R é expressa em bit/amostra. Em se

tratando de codificação de imagens (*e. g.* [15, 16]), R é expressa em bits por *pixel* (bpp).

O objetivo das técnicas de projeto de dicionário é reduzir (para uma determinada taxa R) a distorção introduzida ao se representarem os vetores de entrada $\boldsymbol{x}$ por suas correspondentes versões quantizadas $Q(\boldsymbol{x})$.

3. Algoritmo LBG

Seja a iteração do algoritmo LBG denotada por n. Dados K, N e um limiar de distorção $\epsilon \geq 0$, o algoritmo LBG [3], também conhecido como GLA (*generalized Lloyd algorithm*) consiste da seguinte seqüência de passos:

- *Passo 1*) inicialização: dado um dicionário inicial W_0 e um conjunto de treino $\boldsymbol{X} = \{\boldsymbol{x}_m;\ m = 1,\ 2,\ \ldots,\ M_{\text{vet}}\}$, faça $n = 0$ e $D_{-1} = \infty$;
- *Passo 2*) particionamento: dado W_n (dicionário na n-ésima iteração) aloque cada vetor de treino (vetor de entrada) na respectiva classe (célula de Voronoi) segundo o critério do vetor-código mais próximo; calcule a distorção

$$D_n = \sum_{i=1}^{N} \sum_{\boldsymbol{x}_m \in S_i} d(\boldsymbol{x}_m, \boldsymbol{w}_i); \qquad (5)$$

- *Passo 3*) teste de convergência (critério de parada): se $(D_{n-1} - D_n)/D_n \leq \epsilon$ pare, com W_n representando o dicionário final (dicionário projetado); caso contrário, continue;
- *Passo 4*) atualização do dicionário: compute os novos vetores-código como os centróides das classes de vetores; faça $W_{n+1} \leftarrow W_n$; faça $n \leftarrow n + 1$ e vá para o *Passo 2*.

Em essência, no algoritmo LBG a função distorção decresce monotonicamente, uma vez que o dicionário é iterativamente atualizado visando satisfazer as condições de centróide e de vizinho mais próximo. Na dinâmica do algoritmo LBG, a distorção introduzida ao se representarem os vetores do conjunto de treinamento pelos correspondentes vetores-código (centróides) é monitorada a cada iteração. A regra de parada (teste de convergência) do algoritmo baseia-se nessa distorção monitorada – o treinamento do dicionário é encerrado quando $(D_{n-1} - D_n)/D_n \leq \epsilon$. Existem alguns problemas apresentados pelo algoritmo LBG, comumente relatados [17]: alguns vetores-código podem ser sub-utilizados e, em casos extremos, até mesmo nunca serem utilizados, ou seja, o projeto do dicionário pode resultar em células de Voronoi pequenas ou até mesmo vazias; a velocidade de convergência e o desempenho do dicionário final dependem do dicionário inicial.

4. Algoritmo Competitivo

Na aprendizagem competitiva simples aplicada ao projeto de dicionários, a cada apresentação de um vetor de treino apenas o *vencedor* (isto é, o vetor-peso mais semelhante ao vetor de treino) tem suas componentes atualizadas.

Após a inicialização dos pesos (isto é, das componentes dos N vetores-código K-dimensionais), dicionários podem projetados utilizando o algoritmo competitivo descrito a seguir.

Algoritmo Competitivo (AC):

Para $1 \leq n \leq n_{\text{tot}}$

Para $1 \leq m \leq M_{\text{vet}}$

Determine o vencedor $\boldsymbol{w}_{i^*}(n, m)$:

$$i^* = \arg\min_i d[\boldsymbol{x}(m), \boldsymbol{w}_i(n, m)]$$

Atualize o vencedor de acordo com

$$\tilde{w}_{i^*j}(n, m) = w_{i^*j}(n, m) + \Delta w_{i^*j}(n, m), \qquad (6)$$

em que

$$\Delta w_{i^*j}(n, m) = \eta(n)[(x_j(m) - w_{i^*j}(n, m)]. \qquad (7)$$

Na descrição acima, $\boldsymbol{x}(m)$ é o m-ésimo vetor do conjunto de treino[1], enquanto $\boldsymbol{w}_i(n, m)$ e $\boldsymbol{w}_{i^*}(n, m)$ denotam, respectivamente, o i-ésimo vetor-código e o vencedor quando da apresentação do m-ésimo vetor de treino na n-ésima iteração. Por sua vez,

$$d[\boldsymbol{x}(m), \boldsymbol{w}_i(n, m)] = \sum_{j=1}^{K} [x_j(m) - w_{ij}(n, m)]^2 \qquad (8)$$

denota a distância euclidiana quadrática entre os vetores $\boldsymbol{x}(m)$ e $\boldsymbol{w}_i(n, m)$, em que $x_j(m)$ é a j-ésima componente do vetor $\boldsymbol{x}(m)$ e $w_{ij}(n, m)$ é a j-ésima componente do vetor $\boldsymbol{w}_i(n, m)$. Na expressão que descreve a atualização do vencedor, Δw_{i^*j} é a modificação introduzida na j-ésima componente do vencedor, $\eta(n)$ é a taxa de aprendizagem ou ganho de adaptação na n-ésima iteração (com $0 < \eta(n) < 1$), w_{i^*j} é a j-ésima componente do vencedor e $\tilde{w}_{i^*j}$ é a versão atualizada da j-ésima componente do vencedor[2].

A função taxa de aprendizagem decresce linearmente com n, permanecendo constante ao longo de cada iteração.

O algoritmo AC apresenta 5 parâmetros ajustáveis: dimensão do dicionário (K); número de vetores-código (N); número total de iterações (n_{tot})[3]; taxa de aprendizagem inicial ($\eta(1)$); taxa de aprendizagem final ($\eta(n_{\text{tot}})$).

[1] É importante observar que, por questões de conveniência de notação, utilizou-se $\boldsymbol{x}(m)$ na descrição do algoritmo AC e $\boldsymbol{x}_m$ na descrição do algoritmo LBG.

[2] Note que se omitiram n e m de $\Delta w_{i^*j}(n, m)$, de $w_{i^*j}(n, m)$ e de $\tilde{w}_{i^*j}(n, m)$ para simplificar a notação.

[3] Portanto, em projeto de dicionários utilizando o algoritmo AC, n_{tot} é especificado *a priori*. Por outro lado, em se tratando do algoritmo LBG, o número total de iterações (n_{tot}) depende do parâmetro de convergência ϵ bem como do dicionário inicial utilizados.

5. Avaliação de Complexidade

Neste trabalho, em se tratando do algoritmo LBG, a distância entre um determinado vetor de treino $\boldsymbol{x}$ e o vetor-código $\boldsymbol{w}_i$ é avaliada por meio da distorção (distância euclidiana quadrática)

$$d(\boldsymbol{x}, \boldsymbol{w}_i) = \sum_{j=1}^{K} (x_j - w_{ij})^2, \tag{9}$$

em que w_{ij} é a j-ésima componente do vetor-código $\boldsymbol{w}_i$ e x_j é a j-ésima componente do vetor de treino $\boldsymbol{x}$.

A seguir é apresentada uma avaliação de complexidade dos algoritmos LBG e AC.

5.1 Algoritmo LBG

No *Passo 2* do algoritmo LBG, para que seja determinada a classe a que pertence um vetor de treino, é necessário encontrar a distância entre este vetor e cada um dos N vetores-código do dicionário e depois comparar as distâncias de modo a encontrar o vetor-código mais semelhante: um vetor de treino $\boldsymbol{x}$ é alocado para a i-ésima classe (célula de Voronoi S_i) se $d(\boldsymbol{x}, \boldsymbol{w}_i) < d(\boldsymbol{x}, \boldsymbol{w}_j),\ \forall j \neq i$. A determinação da classe a que pertence um vetor de entrada (vetor de treino) requer, portanto, N cálculos de distância e $N-1$ comparações. Ao ser utilizada a distância euclidiana, cada cálculo de distância requer K multiplicações, K subtrações e $K-1$ adições. A alocação de um vetor de entrada em uma das N classes requer, portanto, KN multiplicações, KN subtrações, $(K-1)N$ adições e $N-1$ comparações. Tendo em vista que o número de vetores de treino é M_{vet}, a cada iteração do algoritmo LBG o processo de alocação dos vetores de treino nas respectivas classes requer KNM_{vet} multiplicações, KNM_{vet} subtrações, $(K-1)NM_{\text{vet}}$ adições e $(N-1)M_{\text{vet}}$ comparações. Em n_{tot} iterações, este processo de alocação no *Passo 2* do algoritmo LBG requer $KNM_{\text{vet}}n_{\text{tot}}$ multiplicações, $KNM_{\text{vet}}n_{\text{tot}}$ subtrações, $(K-1)NM_{\text{vet}}n_{\text{tot}}$ adições e $(N-1)M_{\text{vet}}n_{\text{tot}}$ comparações.

O número de operações requeridas para a determinação de D_n no *Passo 2* do algoritmo LBG é determinado a seguir. A distorção D_n é calculada somando as distâncias entre cada vetor de treino e o correspondente vetor-código da classe à qual foi alocado, o que requer $M_{\text{vet}} - 1$ somas de distâncias. Considerando n_{tot} iterações, a determinação de D_n corresponde a $(M_{\text{vet}} - 1)n_{\text{tot}}$ adições.

Considere agora o *Passo 3*. O teste de convergência do algoritmo LBG precisa realizar uma comparação de $(D_{n-1} - D_n)/D_n$ com ϵ. Para que a comparação seja realizada, são necessárias uma subtração e uma divisão. Considerando n_{tot} iterações, o número total de operações requeridas para a realização do teste de convergência do algoritmo LBG corresponde, portanto, a n_{tot} comparações, n_{tot} subtrações e n_{tot} divisões.

O centróide da i-ésima ($1 \leq i \leq N$) classe é um vetor $\widetilde{\boldsymbol{w}}_i$ tal que sua j-ésima componente corresponde à média aritmética das j-ésimas componentes de todos vetores de treino alocados (no *Passo 2*) em S_i. Seja M_i o número de vetores de treino alocados em S_i. Assim,

$$\sum_{i=1}^{N} M_i = M_{\text{vet}}. \tag{10}$$

A j-ésima ($1 \leq j \leq K$) componente $\widetilde{w}_{ij}$ do centróide $\widetilde{\boldsymbol{w}}_i$ é dada pela média

$$\widetilde{w}_{ij} = \frac{1}{M_i} \sum_{\boldsymbol{x}_m \in S_i} x_{mj}, \tag{11}$$

em que x_{mj} representa a j-ésima componente do vetor de treino $\boldsymbol{x}_m$. A determinação do centróide da i-ésima classe requer K cálculos de média de componentes (cada vetor tem K componentes), o que requer $K(M_i - 1)$ adições e K divisões. O número total de adições para a determinação dos N centróides é

$$\sum_{i=1}^{N} K(M_i - 1) = K(\sum_{i=1}^{N} M_i - \sum_{i=1}^{N} 1). \tag{12}$$

Pela Equação (10) segue que para cada iteração do algoritmo LBG, o *Passo 4* requer $K(M_{\text{vet}} - N)$ adições. O número total de divisões no *Passo 4*, por sua vez, é KN. Admitindo n_{tot} iterações, o número total de operações do *Passo 4* é igual a $K(M_{\text{vet}} - N)n_{\text{tot}}$ adições e KNn_{tot} divisões. Além destas operações, é importante ressaltar que o número (M_i, necessário para cálculo do centróide) de vetores de treino alocados para a i-ésima classe é determinado mediante uso de um contador[4], o que implica M_i adições. Considerando as N classes, são utilizados N contadores (um para cada classe), que consomem $\sum_{i=1}^{N} M_i = M_{\text{vet}}$ adições. Portanto, ao final de n_{tot} iterações, o *Passo 4* do algoritmo LBG requer, também, $M_{\text{vet}}n_{\text{tot}}$ adições.

5.2 Algoritmo Competitivo

No algoritmo competitivo o número de operações requeridas para a determinação do vencedor, isto é, para a determinação de $\boldsymbol{w}_{i^*} : d(\boldsymbol{x}, \boldsymbol{w}_{i^*}) < d(\boldsymbol{x}, \boldsymbol{w}_j), \forall j \neq i^*$, é igual a $KNM_{\text{vet}}n_{\text{tot}}$ multiplicações, $KNM_{\text{vet}}n_{\text{tot}}$ subtrações, $(K-1)NM_{\text{vet}}n_{\text{tot}}$ adições e $(N-1)M_{\text{vet}}n_{\text{tot}}$ comparações.

Para cada vetor de treino, o vencedor (vetor-código mais semelhante a $\boldsymbol{x}$ segundo o critério de distância mínima) é atualizado de acordo com as Equações (6) e (7). Esta atualização requer K adições (uma adição para cada componente de vetor, conforme Equação (6)), K multiplicações (uma multiplicação para cada componente de vetor, conforme Equação (7)) e K subtrações (uma subtração para cada componente de vetor, conforme Equação (7)). Ao final de n_{tot} iterações e considerando M_{vet} vetores de treino, a atualização dos vencedores

[4] Para ser mais preciso, esta contagem é feita no *Passo 2*, na etapa de alocação dos vetores de treino nas respectivas classes.

requer $KM_{\text{vet}}n_{\text{tot}}$ adições, $KM_{\text{vet}}n_{\text{tot}}$ multiplicações e $KM_{\text{vet}}n_{\text{tot}}$ subtrações.

Ao final de cada iteração, a taxa de aprendizagem é calculada de acordo com

$$\eta(n) = \eta(1) + (n-1)\frac{\eta(n_{\text{tot}}) - \eta(1)}{n_{\text{tot}} - 1}. \qquad (13)$$

Observe que o cálculo da taxa de aprendizagem não é realizado ao final da última iteração. Vale lembrar que $\eta(1)$ e $\eta(n_{\text{tot}})$ são parâmetros do algoritmo AC. O valor $\frac{\eta(n_{\text{tot}})-\eta(1)}{n_{\text{tot}}-1}$ é calculado uma única vez (requerendo duas subtrações e uma divisão), ao final da primeira iteração, sendo armazenado para utilização ao final das demais iterações. Ao final de cada iteração (com exceção da última), para que a taxa de aprendizagem seja calculada são necessárias uma adição, uma subtração e uma multiplicação $\left(\text{por } \frac{\eta(n_{\text{tot}})-\eta(1)}{n_{\text{tot}}-1}\right)$. Assim, o número total de operações envolvidas nos $n_{\text{tot}} - 1$ cálculos de $\eta(n)$ corresponde a $2 + n_{\text{tot}} - 1$ subtrações, uma divisão, $n_{\text{tot}} - 1$ adições e $n_{\text{tot}} - 1$ multiplicações.

5.3 LBG versus AC

As Tabelas 1 e 2 apresentam um resumo do número de adições (ad.), subtrações (sub.), divisões (div.), multiplicações (mult.) e comparações (comp.) requeridas pelos algoritmo LBG e AC, respectivamente.

Tabela 1: Número de operações requeridas pelo algoritmo LBG.

	Passo 2
mult.	$KNM_{\text{vet}}n_{\text{tot}}$
sub.	$KNM_{\text{vet}}n_{\text{tot}}$
ad.	$[(K-1)NM_{\text{vet}} + (M_{\text{vet}} - 1)]n_{\text{tot}}$
comp.	$(N-1)M_{\text{vet}}n_{\text{tot}}$
	Passo 3
comp.	n_{tot}
sub.	n_{tot}
div.	n_{tot}
	Passo 4
ad.	$[(K+1)M_{\text{vet}} - KN]n_{\text{tot}}$
div.	KNn_{tot}

A Tabela 3 apresenta o número total de operações dos algoritmos LBG e AC.

Na seção a seguir são apresentados resultados referentes ao projeto de dicionários aplicados à codificação de imagem baseada em QV.

6. Resultados

Os resultados apresentados nesta seção foram obtidos com dicionários projetados utilizando um conjunto de treino constituído de 7 imagens (256×256 *pixels*). Foi considerada quantização vetorial com dimensão $K = 16$, usando em blocos de 4×4 *pixels*. Utilizou-se, portanto,

Tabela 2: Número de operações requeridas pelo algoritmo AC.

	Determinação do vencedor
mult.	$KNM_{\text{vet}}n_{\text{tot}}$
sub.	$KNM_{\text{vet}}n_{\text{tot}}$
ad.	$(K-1)NM_{\text{vet}}n_{\text{tot}}$
comp.	$(N-1)M_{\text{vet}}n_{\text{tot}}$
	Atualização do vencedor
ad.	$KM_{\text{vet}}n_{\text{tot}}$
mult.	$KM_{\text{vet}}n_{\text{tot}}$
sub.	$KM_{\text{vet}}n_{\text{tot}}$
	Cálculo da taxa de aprendizagem
sub.	$n_{\text{tot}} + 1$
div.	1
ad.	$n_{\text{tot}} - 1$
mult.	$n_{\text{tot}} - 1$

um conjunto de treino com 28672 vetores de dimensão 16. Foram projetados dicionários com $N = 32$, 64, 128, 256 e 512 vetores-código. Foram consideradas, portanto, as respectivas taxas de codificação $R = 0{,}3125$ bpp, 0,375 bpp, 0,4375 bpp, 0,5 bpp e 0,5625 bpp. O algoritmo LBG foi executado até que que a modificação na distorção introduzida ao se representar os vetores de treino pelo dicionário fosse inferior a 0,1% ($\epsilon = 0,001$).

As Tabelas 4 e 5 apresentam o número total de iterações dos algoritmos e o correspondente número total de operações dos algoritmos LBG e AC obtidos com as condições (parâmetros, inicialização) que levaram aos dicionários de melhor qualidade, ou seja, aqueles que resultaram nos maiores valores de relação sinal-ruído de pico (PSNR – *Peak Signal-to-noise Ratio*) das imagens reconstruídas. Em se tratando do algoritmo LBG, para cada N, foram testados 5 dicionários iniciais diferentes. Os valores de n_{tot} da Tabela 4 correspondem ao dicionário inicial que levou aos melhores dicionários em termos de PSNR. Conforme mencionado anteriormente, o desempenho do algoritmo LBG e sua velocidade de convergência (número total de iterações) dependem do dicionário inicial: para $N = 128$, por exemplo, os 5 dicionários iniciais diferentes levaram a 43, 59, 17, 26 e 14 iterações.

As Tabelas 4 e 5 mostram que para todos os valores de tamanho do dicionário (N) avaliados, o algoritmo AC requer um número de iterações inferior ao requerido pelo algoritmo LBG: para $N = 256$, por exemplo, o dicionário LBG realiza 15 iterações, enquanto o algoritmo AC requer 4 iterações para projetar o dicionário. Observa-se também nas Tabelas 4 e 5 que o algoritmo AC realiza um número de operações inferior ao realizado pelo algoritmo LBG.

A Figura 1 apresenta o desempenho dos algoritmos LBG e AC na codificação da imagem Mandrill, apresentada na Figura 2. Conforme se pode observar na Figura 1, apesar de o algoritmo competitivo utilizar um número de

Tabela 3: Número total de operações requeridas pelos algoritmos LBG e AC.

	LBG	AC
mult.	$KNM_{\text{vet}}n_{\text{tot}}$	$[1+(1+N)KM_{\text{vet}}]n_{\text{tot}}-1$
sub.	$(1+KNM_{\text{vet}})n_{\text{tot}}$	$[1+(1+N)KM_{\text{vet}}]n_{\text{tot}}+1$
ad.	$[(1+KN)(M_{\text{vet}}-1)+(K-N+1)M_{\text{vet}}]n_{\text{tot}}$	$[1+(K-1)NM_{\text{vet}}+KM_{\text{vet}}]n_{\text{tot}}-1$
div.	$(1+KN)n_{\text{tot}}$	1
comp.	$[1+(N-1)M_{\text{vet}}]n_{\text{tot}}$	$(N-1)M_{\text{vet}}n_{\text{tot}}$

Tabela 4: Número de operações requeridas pelo algoritmo LBG ao serem projetados dicionários destinados à codificação de imagem baseada em QV.

N	R	n_{tot}	mult.	sub.	ad.	div.	comp.
32	0,3125	18	$2,64\cdot10^8$	$2,64\cdot10^8$	$2,57\cdot10^8$	$9,23\cdot10^3$	$1,60\cdot10^7$
64	0,375	13	$3,82\cdot10^8$	$3,82\cdot10^8$	$3,65\cdot10^8$	$1,33\cdot10^4$	$2,35\cdot10^7$
128	0,4375	17	$9,98\cdot10^8$	$9,98\cdot10^8$	$9,45\cdot10^8$	$3,48\cdot10^4$	$6,19\cdot10^7$
256	0,5	15	$1,76\cdot10^9$	$1,76\cdot10^9$	$1,66\cdot10^9$	$6,15\cdot10^4$	$1,10\cdot10^8$
512	0,5625	15	$3,52\cdot10^9$	$3,52\cdot10^9$	$3,31\cdot10^9$	$1,23\cdot10^5$	$2,20\cdot10^8$

Tabela 5: Número de operações requeridas pelo algoritmo AC ao serem projetados dicionários destinados à codificação de imagem baseada em QV.

N	R	n_{tot}	mult.	sub.	ad.	div.	comp.
32	0,3125	3	$4,54\cdot10^7$	$4,54\cdot10^7$	$4,27\cdot10^7$	1	$2,67\cdot10^6$
64	0,375	3	$8,95\cdot10^7$	$8,95\cdot10^7$	$8,40\cdot10^7$	1	$5,42\cdot10^6$
128	0,4375	3	$1,78\cdot10^8$	$1,78\cdot10^8$	$1,67\cdot10^8$	1	$1,09\cdot10^7$
256	0,5	4	$4,72\cdot10^8$	$4,72\cdot10^8$	$4,42\cdot10^8$	1	$2,92\cdot10^7$
512	0,5625	5	$1,18\cdot10^9$	$1,18\cdot10^9$	$1,10\cdot10^9$	1	$7,33\cdot10^7$

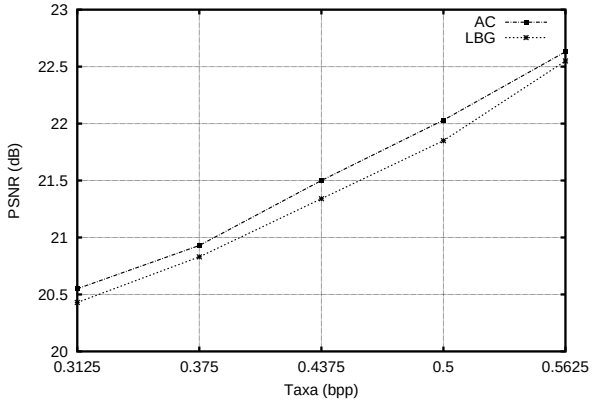

Figura 1: Desempenho dos algoritmos AC e LBG em QV de imagem: PSNR da imagem Mandrill reconstruída versus taxa de codificação para QV com dimensão $K = 16$.

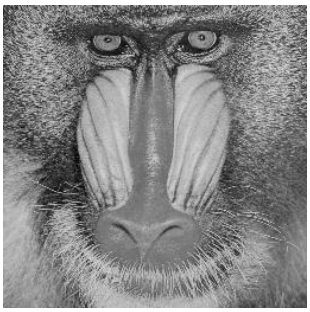

Figura 2: Imagem Mandrill.

operações inferior ao utilizado pelo algoritmo LBG, os dicionários obtidos com o algoritmo competitivo levam a imagens reconstruídas com valores de PSNR ligeiramente superiores (em torno de 0,1 dB a 0,2 dB) aos apresentados pelas imagens reconstruídas obtidas com uso de dicionários LBG.

É oportuno mencionar que a superioridade do algoritmo AC sobre o algoritmo LBG, em termos de número de iterações e número total de operações também foi constatada ao serem utilizados conjuntos de treino com outros tamanhos. Além disso, vale ressaltar que o algoritmo AC requer menos iterações e operações que o algoritmo LBG não apenas em projeto de dicionários aplicados a QV de imagem. Em se tratando de QV de forma de onda de voz o algoritmo AC produz dicionários com número de iterações e número de operações inferiores aos obtidos com o algoritmo LBG e, também, em geral, com qualidade próxima ou um pouco superior à apresentada pelos dicionários LBG.

7. Conclusões

Este trabalho abordou a complexidade computacional de um algoritmo competitivo aplicado ao projeto de quantizadores vetoriais. Foram obtidas expressões para o número de operações (multiplicações, divisões, adições, comparações e subtrações) realizadas pelo algoritmo competivo e pelo tradicional algoritmo LBG, em função da dimensão dos vetores-código, do tamanho do dicionário, do número de vetores do conjunto de treino e do número de iterações (passagens completas do conjunto de treino) realizadas. Foram apresentados resultados referentes ao projeto de dicionários aplicados à codificação de imagem. Nas avaliações realizadas observou-se que o algoritmo competitivo requer um número de operações inferior ao requerido pelo algoritmo LBG. As simulações realizadas mostraram que um melhor desempenho em termos de PSNR das imagens reconstruídas pode ser obtido com o uso de dicionários projetados com o algoritmo competitivo em substituição aos dicionários LBG.

Referências

[1] R. M. Gray. Vector quantization. *IEEE ASSP Magazine*, pages 4–29, April 1984.

[2] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Boston, MA, 1992.

[3] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1):84–95, January 1980.

[4] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, September 1990.

[5] S. Carrato. Image vector quantization using ordered codebooks: Properties and applications. *Signal Processing*, 40:87–103, 1994.

[6] A. K. Krishnamurthy, S. C. Ahalt, D. E. Melton, and P. Chen. Neural networks for vector quantization of speech and images. *IEEE Journal on Selected Areas in Communications*, 8(8):1449–1457, October 1990.

[7] O. T.-C. Chen, B. J. Sheu, and W.-C. Fang. Image compression using self-organization networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 4(5):480–489, October 1994.

[8] C. Zhu and L. M. Po. Partial distortion sensitive competitive learning algorithm for optimal codebook design. *Electronics Letters*, 32(19):1757–1758, September 1996.

[9] E. Yair, K. Zeger, and A. Gersho. Competitive learning and soft competition for vector quantizer design. *IEEE Transactions on Signal Processing*, 40(2):294–309, February 1992.

[10] K. Zeger, J. Vaisey, and A. Gersho. Globally optimal vector quantizer design by stochastic relaxation. *IEEE Transactions on Signal Processing*, 40(2):310–322, February 1992.

[11] N. B. Karayiannis and P.-I. Pai. Fuzzy vector quantization algorithms and their applications in image compression. *IEEE Transactions on Image Processing*, 4(9):1193–1201, September 1995.

[12] N. B. Karayiannis, P.-I. Pai, and N. Zervos. Image compression based on fuzzy algorithms for learning vector quantization and wavelet image decomposition. *IEEE Transactions on Image Processing*, 7(8):1223–1230, August 1998.

[13] J. S. Pan, F. R. McInnes, and M. A. Jack. VQ codebook design using genetic algorithms. *Electronics Letters*, 31(17):1418–1419, August 1995.

[14] A. Gersho and V. Cuperman. Vector quantization: A pattern-matching technique for speech coding. *IEEE Communications Magazine*, pages 15–20, December 1983.

[15] B. Ramamurthi and A. Gersho. Classified vector quantization of images. *IEEE Transactions on Communications*, 34(11):1105–1115, November 1986.

[16] N. M. Nasrabadi and R. A. King. Image coding using vector quantization: A review. *IEEE Transactions on Communications*, 36(8):957–971, August 1988.

[17] D. Lee, S. Baek, and K. Sung. Modified K-means algorithm for vector quantizer design. *IEEE Signal Processing Letters*, 4(1):2–4, January 1997.