TEMA Tend. Mat. Apl. Comput., **13**, No. 2 (2012), 193-206.
doi: 10.5540/tema.2012.013.02.0193

Uma Alternativa de Aceleração do Algoritmo *Fuzzy* K*-Means* Aplicado à Quantização Vetorial

F. MADEIRO[1], R.R.A. GALVÃO[2], Escola Politécnica de Pernambuco, Universidade de Pernambuco, 50720-001 Recife, PE, Brasil.

F.A.B.S. FERREIRA[3], Instituto de Estudos Avançados em Comunicações, 58429-900 Campina Grande, PB, Brasil.

D.C. CUNHA[4], Centro de Informática, Universidade Federal de Pernambuco, 50740-560 Recife, PE, Brasil.

Resumo. Compressão de sinais, marca d´água digital e reconhecimento de padrões são exemplos de aplicações de quantização vetorial (QV). Um problema relevante em QV é o projeto de dicionários. Neste trabalho, é apresentada uma alternativa de aceleração do algoritmo *fuzzy* K*-Means* aplicado ao projeto de dicionários. Resultados de simulações envolvendo QV de imagens e de sinais com distribuição de Gauss-Markov mostram que o método proposto leva a um aumento da velocidade de convergência (redução do número de iterações) do algoritmo *fuzzy* K*-Means* sem comprometimento da qualidade dos dicionários projetados.

Palavras-chave. Quantização vetorial, *fuzzy* K*-Means*, codificação de imagens.

1. Introdução

A quantização é uma técnica relevante em sistemas de compressão de sinais [12, 21]. Seu alvo é a digitalização (discretização) dos valores de amostras de um sinal. Há duas classes gerais de quantização: escalar e vetorial. A primeira consiste em um mapeamento $Q : \mathbb{R} \rightarrow C$, em que C é um subconjunto do espaço Euclidiano unidimensional $\mathbb{R}$. O número de elementos de C é denominado número de níveis do quantizador, denotado por L. Ao final do processo de quantização, cada amostra poderá ser codificada, por exemplo, em uma palavra-binária de $l = \log_2 L$ bits. A quantização vetorial (QV), por sua vez, consiste em um mapeamento $Q : \mathbb{R}^N \rightarrow W$, em que $W = \{\vec{w}_i;\ i = 1,\ 2,\ \ldots,\ K\} \subset \mathbb{R}^N$ é denominado dicionário, N é a dimensão do quantizador vetorial e K é o tamanho do dicionário (número de vetores-código). A taxa de codificação da quantização vetorial é dada por $\frac{1}{N} \log_2 K$, expressa em bits por amostra em codificação de forma de onda de voz e em bits por *pixel* (bpp) em codificação de imagem.

[1]madeiro@poli.br
[2]rodrigoregisag@gmail.com.
[3]felipebsferreira@gmail.com
[4]dcunha@cin.ufpe.br

Recebido em 05 Agosto 2012; Aceito em 20 Agosto 2012.

A Figura 1 ilustra a QV de imagem, realizada no domínio dos *pixels* (domínio original, isto é, sem uso de transformadas). Observa-se que a imagem é dividida em blocos de *pixels*. No exemplo, são usados blocos de 4×4 *pixels* (dimensão $N = 16$) e um dicionário de tamanho $K = 32$. O processo de quantização consiste em substituir cada bloco de *pixels* da imagem original pelo vetor-código (bloco de *pixel*) mais próximo (segundo um critério de distância) dentre os 32 vetores-código do dicionário. A qualidade da imagem quantizada depende do dicionário utilizado. Em outras palavras, dados dois dicionários de mesmo tamanho e mesma dimensão, diz-se que um deles tem melhor qualidade quando leva à imagem quantizada com qualidade superior (segundo um critério de distorção) à obtida com uso do outro dicionário.

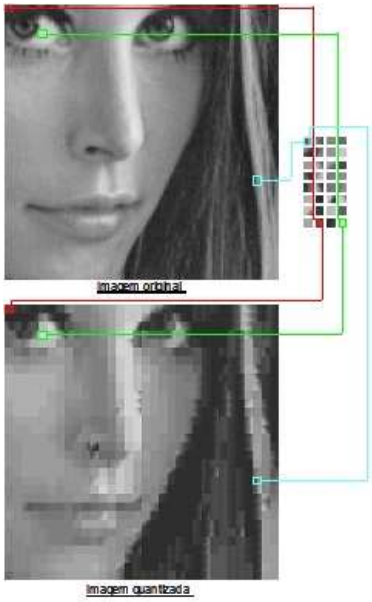


Figura 1: Exemplo de QV de imagem.

Dentre os desafios existentes em QV, podem ser citados: o projeto de dicionários [2], a complexidade computacional [20] e a sensibilidade da técnica aos erros de transmissão [17]. Ressalte-se que, além da compressão de sinais, há um amplo espectro de aplicações para QV, como, por exemplo, esteganografia [7, 8], marca d'água digital [9, 18], identificação vocal [25] e classificação de sinais de voz com patologias [27].

Como técnicas para projeto de dicionários, podem ser citadas: algoritmo LBG (Linde-Buzo-Gray) [19]; algoritmos *fuzzy* [26] e algoritmos de aprendizagem competitiva [5] e algoritmos meméticos [3]. Em se tratando de QV de imagem, o alvo das técnicas é minimizar a distorção introduzida no processo de substituição dos blocos de pixels originais pelos respectivos vetores-código. No que diz respeito à QV de sinais unidimensionais, como é o caso dos sinais de voz, o alvo é minimizar a distorção introduzida ao se substituírem os blocos de amostras originais pelos respectivos vetores-código.

Neste trabalho, é apresentada uma técnica de aceleração do algoritmo *fuzzy* K*-Means*

aplicado ao projeto de dicionários. Resultados de simulação concernentes à QV de imagens e de sinais com distribuição de Gauss-Markov mostram que a técnica apresentada leva a uma redução do número de iterações realizadas no projeto de dicionários, sem que haja comprometimento da qualidade dos dicionários obtidos. Convém mencionar que na literatura podem ser encontrados diversos trabalhos envolvendo lógica nebulosa (*fuzzy*) no âmbito de QV, e.g. [6, 11, 13, 23, 24].

2. Algoritmo K-*Means*

O projeto de dicionários possui grande importância para o bom desempenho de sistemas de compressão de sinais baseados em QV. De uma maneira geral, o projeto de dicionários é um processo de agrupamento de dados (ou *clusterização*) [10]. Tal processo consiste em agrupar um conjunto de dados (M blocos ou vetores de amostras extraídos do conjunto de treino) em grupos ou *clusters* (K células de quantização, tal que $K < M$), de maneira que vetores do mesmo grupo apresentem alta similaridade entre si e possuam pouca similaridade com vetores de outros grupos.

Existem muitas técnicas para agrupamento de dados na literatura. Em [28], elas são classificadas como métodos hierárquicos e métodos de partição. Os métodos de partição são subdivididos em métodos de *clusterização* abrupta (*hard*) e *clusterização fuzzy*. No primeiro caso, podemos citar o algoritmo K-*Means* [12], em que cada vetor de treino (bloco de amostras do conjunto de dados original) pertence a uma e apenas uma célula de quantização. Já no caso da técnica baseada em *clusterização fuzzy*, na qual se enquadra, por exemplo, o algoritmo *fuzzy* K-*Means* [15], cada vetor de treino pode estar associado a várias células de quantização, com um certo grau de pertinência a cada célula.

O algoritmo K-*Means* é brevemente apresentado a seguir.

Seja $X = \{\vec{x}_j\}, j = 1, 2, ..., M$, um conjunto de treino composto por M vetores N-dimensionais, com $M \gg K$. O algoritmo K-*Means* particiona o espaço vetorial $\mathbb{R}^N$ atribuindo cada vetor de treino a um único *cluster* através da busca do vizinho mais próximo (VMP). Precisamente, $\vec{x}_j$ pertencerá ao *cluster* (ou região de Voronoi ou célula de quantização) $V(\vec{w}_i)$ se $d(\vec{x}_j, \vec{w}_i) < d(\vec{x}_j, \vec{w}_a), \forall a \neq i$, em que $d(\vec{x}_j, \vec{w}_i)$ denota a distância Euclidiana quadrática entre $\vec{x}_j$ e $\vec{w}_i$. Neste caso, diz-se que $\vec{w}_i$ é o VMP de $\vec{x}_j$. Pode-se associar a busca do VMP a uma função de pertinência, definida por

$$\mu_i(\vec{x}_j) = \begin{cases} 1, & se\ \vec{w}_i = \mathrm{VMP}(\vec{x}_j) \\ 0, & \text{caso contrário} \end{cases} . \tag{2.1}$$

Dessa forma, a distorção obtida ao se representarem todos os vetores do conjunto de treino pelos respectivos VMPs é dada por

$$J_1 = \sum_{i=1}^{K} \sum_{j=1}^{M} \mu_i(\vec{x}_j) d(\vec{x}_j, \vec{w}_i). \tag{2.2}$$

Para minimizar J_1, os vetores $\vec{w}_i$ são atualizados como segue:

$$\vec{w}_i = \frac{\sum_{j=1}^{M} \mu_i(\vec{x}_j)\vec{x}_j}{\sum_{j=1}^{M} \mu_i(\vec{x}_j)},\ i = 1, 2, ..., K. \tag{2.3}$$

Depois de inicializado o conjunto de vetores $\vec{w}_i, i = 1, 2, ..., K$, o algoritmo K-*Means* aplicado ao projeto de dicionários (quantizadores vetoriais) pode ser assim resumido:

1. Particionamento – o conjunto de treino é particionado em K *clusters* de acordo com a regra do VMP.
2. Atualização do dicionário – os novos vetores-código são os centróides dos *clusters*, calculados de acordo com a Equação (2.3).
3. Teste de convergência – critério de parada do algoritmo.

As etapas de particionamento e atualização são realizadas até que o critério de parada seja satisfeito. Precisamente, o algoritmo pára ao final da t-ésima iteração se

$$\frac{J_1(t-1) - J_1(t)}{J_1(t)} \leq \epsilon, \tag{2.4}$$

em que ϵ é um parâmetro do algoritmo, denominado limiar de distorção, e $J_1(t)$ denota a distorção obtida no particionamento da t-ésima iteração.

3. Algoritmo *fuzzy* K*-Means*

O algoritmo *fuzzy* K*-Means* tem por objetivo minimizar a distorção entre os vetores de treinamento $\vec{x}_j$ e os vetores-código $\vec{w}_i$ que compõem o dicionário. A medida de distorção é definida em [4] e dada por

$$J_m = \sum_{i=1}^{K} \sum_{j=1}^{M} \mu_i \left(\vec{x}_j\right)^m d\left(\vec{x}_j, \vec{w}_i\right) \ , 1 < m < \infty \tag{3.5}$$

em que $\mu_i(\vec{x}_j)$ é uma função que representa o grau de pertinência do vetor $\vec{x}_j$ à i-ésima célula de quantização e m é o parâmetro que controla a nebulosidade da *clusterização*. Para um determinado dicionário e considerando que $\mu_i(\vec{x}_j) \in [0, 1], i = 1, \ldots, K$; $0 < \sum_{j=1}^{M} \mu_i \left(\vec{x}_j\right) < M$ e $\sum_{i=1}^{K} \mu_i \left(\vec{x}_j\right) = 1$, a minimização da função J_m resulta em

$$\mu_i \left(\vec{x}_j\right) = \frac{1}{\sum_{k=1}^{K} \left(\frac{d(\vec{x}_j, \vec{w}_i)}{d(\vec{x}_j, \vec{w}_k)}\right)^{\frac{1}{m-1}}} \ . \tag{3.6}$$

Para um dado conjunto de funções de grau de pertinência, os vetores-código do dicionário podem ser obtidos por meio da minimização de $J_m(\vec{w}_i)$, tal que

$$\vec{w}_i = \frac{\sum_{j=1}^{M} \mu_i(\vec{x}_j)^m \vec{x}_j}{\sum_{j=1}^{M} \mu_i(\vec{x}_j)^m} \ , i = 1, 2, \ldots, K. \tag{3.7}$$

3.1. Algoritmo *Fuzzy* K*-Means* acelerado

Um dos desafios em se tratando de métodos de *clusterização* é o aumento da velocidade de convergência (redução do número de iterações). No âmbito da quantização vetorial, algumas alternativas foram propostas para acelerar o algoritmo K*-Means*, tais como as técnicas de Lee *et al*. [16] e de Paliwal-Ramasubramanian [22].

Em ambas as técnicas, ao final de cada iteração, os vetores-código do dicionário são recalculados de acordo com a expressão

$$\vec{w}_i{}^t = \vec{w}_i{}^{t-1} + s\left(\mathcal{C}\left(V\left(\vec{w}_i{}^{t-1}\right)\right) - \vec{w}_i{}^{t-1}\right) \ , \tag{3.8}$$

em que $\vec{w}_i{}^t$ é o vetor-código $\vec{w}_i$ na t-ésima iteração, s é um fator de escala e $\mathcal{C}\left(V\left(\vec{w}_i{}^{t-1}\right)\right)$ é o centróide da partição $V\left(\vec{w}_i{}^{t-1}\right)$. A diferença entre as técnicas de aceleração do algoritmo K*-Means* está no fator de escala s. Enquanto em [16] s assume um valor fixo, na técnica de Paliwal-Ramasubramanian o fator de escala é função da iteração t, o que leva a uma maior velocidade de convergência. Neste caso, o fator de escala s é dado pela equação

$$s = 1 + \frac{x}{(x+t)} \ , \tag{3.9}$$

em que $x > 0$.

Assim, para as versões aceleradas do algoritmo K*-Means*, cada novo vetor-código não é mais determinado como o centróide de sua classe, mas sim por um dos pontos na linha que conecta o vetor-código atual com seu ponto refletido, passando pelo novo centróide da classe, conforme ilustra a Figura 2. O algoritmo K*-Means* determina o ponto 2, correspondente ao novo centróide da classe, como o novo vetor-código. A proposta de Lee *et al*., ao utilizar uma abordagem do tipo *look ahead*, escolhe como novo vetor-código um ponto entre os pontos 2 e 4, visando acelerar a convergência do algoritmo K*-Means*. Em suma, o novo vetor-código é determinado da seguinte forma: *novo vetor-código = vetor-código atual + escala* $\times$ *(novo centróide – vetor-código atual)*.

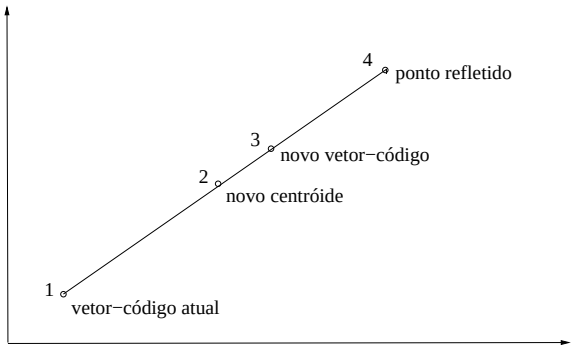


Figura 2: Atualização do dicionário pelo método de Lee *et al*.

Uma inspeção da Equação (3.8) revela que a proposta de redução do número de iterações tem como pressuposto uma tendência de deslocamento dos vetores-código segundo

uma trajetória aproximadamente retilínea. Simulações realizadas com projeto de dicionários para QV de sinais unidimensionais, realizadas pelos autores deste trabalho, mostraram que o pressuposto é satisfeito pelo algoritmo K-*Means* bem como pelo *fuzzy* K-*Means*.

A técnica apresentada em [22], utilizada para acelerar o algoritmo K-*Means*, é aplicada neste trabalho para acelerar o algoritmo *fuzzy* K-*Means*. Pela inspeção da Equação (3.9), percebe-se que o fator de escala s varia lenta e inversamente com a iteração t. A razão de se utilizar um fator de escala variável se justifica pelo fato de que as atualizações devem ser progressivamente menos acentuadas. Com isso, nas iterações próximas à convergência, não é interessante o uso de um fator de escala alto, sob o risco de haver uma perturbação indesejada no dicionário. Por fim, o fator s deve satisfazer a duas condições: (1) $s > 1$, para assegurar uma convergência mais rápida que a do algoritmo *fuzzy* K-*Means*; (2) $s < 2$, para evitar que o algoritmo tenha uma convergência muito lenta ou não convirja.

Neste trabalho, portanto, a versão acelerada do algoritmo *fuzzy* K-*Means* consiste em recalcular os vetores-código do algoritmo *fuzzy* K-*Means* convencional por meio das Equações (3.8) e (3.9).

4. Resultados

Nesta seção, são apresentados resultados obtidos para projeto de dicionários usando como conjunto de treino (i) sinais com distribuição de Gauss-Markov e (ii) imagens digitais. Os algoritmos *fuzzy* K-*Means* (denotado por FKM) e *fuzzy* K-*Means* acelerado (denotado por AFKM) param ao final da t-ésima iteração, se

$$\frac{J_m(t-1) - J_m(t)}{J_m(t)} \leq \epsilon, \tag{4.10}$$

com J_m dado na Equação (3.5), com $m = 5$, em conformidade com as recomendações de [15], e $\epsilon = 10^{-3}$ em todas as simulações realizadas.

Antes, entretanto, é importante mostrar que, a cada iteração, os vetores-código no algoritmo FKM tendem a se deslocar segundo uma trajetória aproximadamente retilínea, conforme mostra a Figura 3, na qual as cores e os pontos indicados representam, respectivamente, regiões de Voronoi e seus centróides. Desta forma, a Figura 3 justifica a proposta de aceleração do presente artigo.

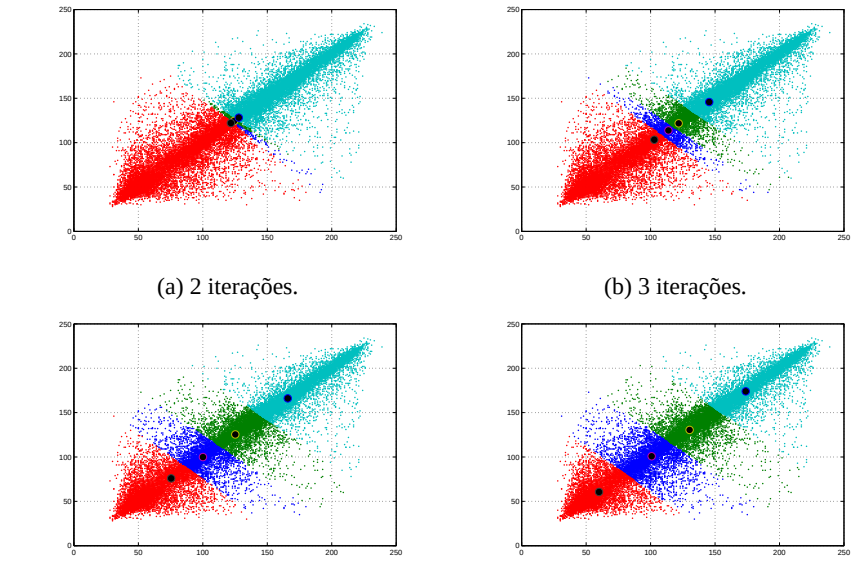
4.1. Fonte de Gauss-Markov

A fonte de Gauss-Markov de 1^a. ordem é também conhecida como fonte de Markov de 1^a. ordem ou fonte AR(1) [14]. Para efeito de simplicidade, essa fonte será referenciada como fonte de Gauss-Markov ao longo de todo o texto.

O processo discreto de Gauss-Markov $\{X(n)\}$ é definido pela equação

$$x(n) = ax(n-1) + w(n), \ \ \forall n \tag{4.11}$$

em que $\{w(n)\}$ é uma sequência de variáveis aleatórias Gaussianas independentes e identicamente distribuídas com média zero e a é um parâmetro denominado coeficiente de correlação, que determina o grau de correlação entre $x(n)$ e $x(n-1)$.



(a) 2 iterações. (b) 3 iterações.

(c) 4 iterações. (d) 5 iterações.

Figura 3: Trajetória de deslocamento retilínea dos vetores-código no algoritmo FKM.

Em todas as simulações realizadas com sinais com distribuição de Gauss-Markov, utilizou-se conjunto de treinamento com 150.000 amostras. Foram projetados dicionários para QV com taxa de 1,0 bit por amostra. Essa taxa é obtida para diversas combinações de dimensão N e número de níveis K, como, por exemplo, $N = 6$ e $K = 64$; $N = 8$ e $K = 256$. Para cada combinação, foram utilizadas 40 inicializações aleatórias diferentes de cada algoritmo. A qualidade dos dicionários projetados foi avaliada por meio da relação sinal-ruído (SNR, *signal-to-noise ratio*), assim definida:

$$\mathrm{SNR} = 10 \log_{10} \left(\frac{\sum\limits_{n} x^2(n)}{\sum\limits_{n} [x(n) - y(n)]^2} \right), \tag{4.12}$$

em que $x(n)$ e $y(n)$ representam, respectivamente, o sinal original e o sinal quantizado no instante de tempo n.

São apresentados resultados para valores médios de SNR dos sinais reconstruídos e número médio de iterações para os dois algoritmos (FKM e AFKM), assim como o número de casos em que o algoritmo AFKM realizou um número menor de iterações (ou seja, teve maior velocidade de convergência) que o algoritmo FKM e o número de casos em que o algoritmo AFKM resultou em sinais reconstruídos com um valor maior de SNR média.

Os resultados mencionados anteriormente estão organizados nas Tabelas 1 e 2, referentes a distribuições de Gauss-Markov com coeficientes de correlação $a = 0,0$ e $a = 0,9$, respectivamente. Observa-se na Tabela 1, por exemplo, para $K = 64$, que o número médio de iterações realizadas pelo algoritmo AFKM é igual a 20,15, ao passo que o número cor-

respondente para o algoritmo FKM é 24,30. Observa-se, ainda, que, dentre as 40 inicializações, em 30 o algoritmo AFKM foi mais rápido (realizou um menor número de iterações) que o FKM; dentre essas 30, em 22 inicializações o algoritmo AFKM levou a sinais reconstruídos com SNR superior à obtida com uso de dicionário FKM. De uma forma geral, a inspeção da Tabela 1 revela que o algoritmo AFKM converge mais rapidamente que o algoritmo FKM (realiza um menor número de iterações), sem que haja comprometimento dos dicionários projetados – de fato, os valores médios de SNR obtidos com dicionários AFKM são, em geral, iguais ou ligeiramente superiores aos obtidos com dicionários FKM. Esse comportamento é também observado na Tabela 2.

Tabela 1: Resultados obtidos para sinais com distribuição de Gauss-Markov com coeficiente de correlação $a = 0,0$.

K	SNR Média		Nº Médio de Iterações		Nº de casos em que	Nº de casos em que
	FKM	**AFKM**	FKM	**AFKM**	AFKM teve menos iterações	AFKM levou a maiores SNRs
4	4,36	**4,38**	8,75	**8,25**	22	16
8	4,44	**4,45**	12,12	**9,97**	30	18
16	4,60	**4,62**	17,46	**14,76**	28	24
32	4,76	**4,79**	21,43	**18,72**	32	24
64	4,95	**4,96**	24,30	**20,15**	30	22
128	5,33	**5,35**	25,60	**21,56**	28	25
256	5,63	**5,63**	26,78	**20,18**	30	24

Tabela 2: Resultados obtidos para sinais com distribuição de Gauss-Markov com coeficiente de correlação $a = 0,9$.

K	SNR Média		Nº Médio de Iterações		Nº de casos em que	Nº de casos em que
	FKM	**AFKM**	FKM	**AFKM**	AFKM teve menos iterações	AFKM levou a maiores SNRs
4	7,96	**7,96**	12,32	**9,17**	31	25
8	9,36	**9,38**	18,95	**14,62**	25	20
16	10,19	**10,20**	26,52	**25,45**	22	15
32	10,70	**10,71**	21,43	**17,15**	30	29
64	11,05	**11,06**	24,30	**19,25**	34	31
128	11,33	**11,34**	20,58	**18,66**	31	27
256	11,91	**11,92**	24,15	**19,92**	30	25

4.2. Imagens

Os resultados apresentados nesta subseção foram obtidos considerando as imagens Elaine, Goldhill, Clock e Lena, todas com 256×256 pixels, 256 níveis de cinza e ilustradas na Figura 4. Foi considerada QV com dimensão $N = 16$, o que corresponde a blocos de 4×4 *pixels*, e dicionários projetados com $K = 32, 64, 128, 256$ e 512 vetores-código. Para cada tamanho de dicionário considerado, foram utilizadas 40 inicializações aleatórias diferentes

de cada algoritmo. A qualidade dos dicionários projetados foi avaliada por meio da relação sinal-ruído de pico (PSNR, *peak signal-to-noise ratio*), assim definida para imagens 256×256 originalmente codificadas a $8,0$ bpp:

$$\text{PSNR (dB)} = 10 \log_{10} \left[\frac{V_p^2}{\text{MSE}} \right], \tag{4.13}$$

em que MSE (*mean square error*) denota o erro médio quadrático entre essas imagens e V_p denota o maior valor de nível de cinza (para o caso de imagens originais 8,0 bpp, tem-se $V_p = 255$).

São apresentados resultados para os mesmos parâmetros considerados na subseção referente a sinais com distribuição de Gauss-Markov. Tais resultados encontram-se organizados nas Tabelas 3 e 4, referentes às imagens Elaine e Goldhill, respectivamente, e na Figura 5, referente à imagem Clock, e devem ser entendidos como segue.

Considere, por exemplo, a Tabela 3. Para o conjunto de treino correspondente à imagem Elaine e assumindo dicionários de tamanho $K = 512$, o valor médio da PSNR das imagens reconstruídas com dicionários obtidos pelo algoritmo FKM foi de $33,12$ dB, enquanto o correspondente valor médio obtido pelos dicionários AFKM foi de $33,18$ dB. Cabe ressaltar que os valores médios de PSNR calculados levam em consideração o uso de 40 inicializações de dicionário distintas, conforme mencionado no início desta subseção. Ainda com relação à Tabela 3 e considerando $K = 512$, o número médio de iterações do algoritmo FKM foi $47,07$, ao passo que o algoritmo AFKM teve uma maior velocidade média de convergência – de fato, o número médio de iterações para o algoritmo AFKM foi $32,06$. Para $K = 512$, a Tabela 3 revela ainda que, das 40 inicializações consideradas, em 36 delas o algoritmo AFKM realizou um menor número de iterações. Por fim, dentre as 36, em 28 os dicionários AFKM levaram a imagens Elaine reconstruídas com valores de PSNR superiores aos obtidos com os dicionários FKM.

Considere agora a Tabela 4. Para $K = 512$, a técnica proposta no presente trabalho contribuiu para reduzir em cerca de 26% o número de iterações do algoritmo *fuzzy* K-*Means*. De fato, o algoritmo FKM tem um número médio de iterações igual a $46,40$, enquanto o AFKM tem um número médio de iterações igual a $34,24$. Para $K = 512$, os valores médios de PSNR das imagens reconstruídas foram bem próximos com o uso de dicionários FKM e AFKM, sendo $30,76$ dB no primeiro caso e $30,82$ dB no segundo.

As Tabelas 3 e 4 mostram que, para cada imagem considerada nas simulações e os diversos valores de K considerados, o algoritmo AFKM possui, em geral, uma maior velocidade de convergência do que aquela apresentada pelo algoritmo FKM. Em outras palavras, para a maioria das 40 inicializações consideradas para cada valor de K, o algoritmo AFKM realiza um menor número de iterações comparado ao algoritmo FKM. Adicionalmente, as tabelas revelam que o aumento de velocidade de convergência não ocorre à custa de um comprometimento de qualidade dos dicionários projetados. Na realidade, o algoritmo AFKM produz, em geral, com um menor número de iterações, dicionários de qualidade similiar ou levemente superior (valores de PSNR praticamente iguais ou ligeiramente superiores) àquela apresentada pelos dicionários FKM.

Conforme se observa na Figura 5, o algoritmo AFKM converge mais rapidamente que o FKM. De fato, o primeiro realiza um menor número de iterações para projetar o dicionário. Além disso, a Figura 5 revela que, fixada uma iteração, o dicionário AFKM tem qualidade

(a) Elaine. (b) Goldhill. (c) Clock. (d) Lena.

Figura 4: Imagens 256×256 usadas nas simulações.

Tabela 3: Resultados obtidos para a imagem Elaine.

K	PSNR Média		N$^{o\cdot}$ Médio de Iterações		N$^{o\cdot}$ de casos em que	N$^{o\cdot}$ de casos em que
	FKM	**AFKM**	FKM	**AFKM**	AFKM teve menos iterações	AFKM levou a maiores PSNRs
32	27,61	**27,63**	29,22	**25,35**	28	20
64	29,01	**29,02**	31,60	**25,05**	29	23
128	30,34	**30,35**	35,35	**26,80**	32	16
256	31,71	**31,76**	38,42	**31,36**	33	24
512	33,12	**33,18**	47,07	**32,06**	36	28

Tabela 4: Resultados obtidos para a imagem Goldhill.

K	PSNR Média		N$^{o\cdot}$ Médio de Iterações		N$^{o\cdot}$ de casos em que	N$^{o\cdot}$ de casos em que
	FKM	**AFKM**	FKM	**AFKM**	AFKM teve menos iterações	AFKM levou a maiores PSNRs
32	26,49	**26,55**	30,62	**25,03**	25	20
64	27,53	**27,52**	28,80	**21,55**	30	25
128	28,45	**28,45**	30,75	**24,85**	32	25
256	29,54	**29,52**	37,02	**27,33**	34	19
512	30,76	**30,82**	46,40	**34,24**	37	27

superior à do dicionário FKM, levando a uma imagem reconstruída com maior valor de PSNR. A Figura 5 também permite observar que uma determinada qualidade de imagem reconstruída (ou, de forma equivalente, uma dada qualidade de dicionário projetado) é obtida mais prematuramente (em um menor número de iterações) pelo algoritmo AFKM.

Finalmente, é válido mencionar que abordagens evolutivas, e.g. [1], podem levar a dicionários de melhor qualidade quando comparados aos obtidos pelo algoritmo AFKM. No entanto, isto ocorre às custas de investimento em operadores genéticos. Para a imagem Lena, por exemplo, para dicionários de tamanhos 32 e 64, há um ganho de 0,25 dB e de 0,28 dB respectivamente ao serem usados dicionários projetados com o algoritmo [1] em substituição aos dicionários AFKM.

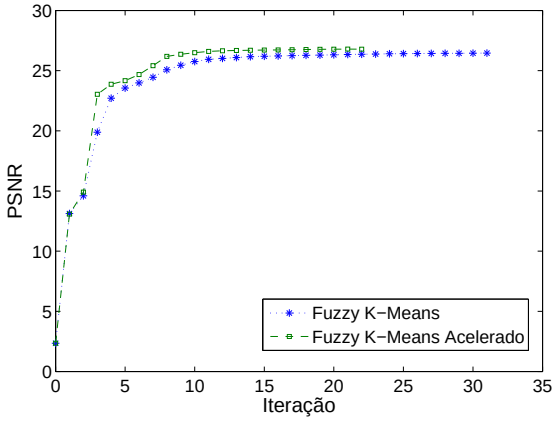


Figura 5: PSNR da imagem Clock reconstruída ao final de cada iteração dos algoritmos FKM e AFKM, considerando a mesma inicialização para ambos os algoritmos.

5. Conclusão

Neste trabalho, foi apresentada uma técnica para aumentar a velocidade de convergência do algoritmo *fuzzy K-Means* aplicado ao projeto de dicionários para compressão de sinais baseada em quantização vetorial. A técnica consiste, essencialmente, em recalcular os vetores-código ao final de cada iteração do algoritmo, por meio de um procedimento que tem como pressuposto o deslocamento dos vetores-código segundo trajetórias aproximadamente retilíneas ao longo das iterações do algoritmo. Por meio de simulações envolvendo o projeto de dicionários para quantização vetorial de sinais com distribuição de Gauss-Markov e imagens digitais, observou-se que a técnica proposta contribui para reduzir o número de iterações realizadas pelo algoritmo *fuzzy K-Means*. Os resultados de simulações mostraram que o aumento da velocidade de convergência do algoritmo não ocorre à custa de prejuízo de qualidade dos dicionários projetados. Em outras palavras, a relação sinal-ruído de pico (PSNR) das imagens reconstruídas com o uso de dicionários projetados pela técnica proposta é praticamente a mesma ou ligeiramente superior à PSNR das imagens reconstruídas com o uso de dicionários projetados pelo versão original do algoritmo *fuzzy K-Means*. Em projetos de dicionários com taxa de 1,0 bit por amostra, tendo como conjunto de treino uma fonte de Gauss-Markov de 1^a. ordem, a técnica apresentada neste trabalho levou a reduções de até cerca de 25% no número médio de iterações.

Agradecimentos

Os autores expressam os agradecimentos ao CNPq pelo apoio financeiro.

Abstract. Signal compression, digital watermarking and pattern recognition are examples of applications of vector quantization (VQ). A relevant problem concerning VQ is codebook design. In this paper, an alternative is presented for accelerating the fuzzy K-it Means algorithm applied to codebook design. Simulation results involving VQ of images and Gauss-Markov

signals show that the proposed method leads to an increase of convergence speed (reduction of the number of iterations) of the fuzzy K-it Means algorithm without sacrificing the quality of the designed codebooks.

Keywords. Vector quantization, *fuzzy K-Means*, image coding.

Referências

[1] C.R.B. Azevedo, T.A.E. Ferreira, W.T.A. Lopes, F. Madeiro, Improving Image Vector Quantization with a Genetic Accelerated K-Means Algorithm, in *Advanced Concepts for Intelligent Vision Systems*, vol. **5259**/2008 of *Lecture Notes in Computer Science*, pp. 67-76, Springer Berlin/Heidelberg, (2008).

[2] C.R.B. Azevedo, E.L.B. Junior, T.A.E. Ferreira, F. Madeiro, M.S. Alencar, An Evolutionary Approach for Vector Quantization Codebook Optimization, *Lectures Notes in Computer Science*, **5263** (2008), 452-461.

[3] C.R.B. Azevedo, F.E.A.G. Azevedo, W.T.A. Lopes, F. Madeiro, Terrain-Based Memetic Algorithms to Vector Quantization Design, in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2008)*, edited by N. Krasnogor, vol. 236 of *Studies in Computational Intelligence*, chapter 17, pp. 197-211. Springer-Verlag, Berlin, 1st edition, (2009), ISBN 978-3-642-03210-3.

[4] J.C. Bezdek, *Pattern Recognition with Objective Function Algorithms*, Plenum, New York, 1981.

[5] E.L. Bispo Junior, C.R.B. Azevedo, W.T.A. Lopes, M.S. Alencar, F. Madeiro, Methods to Accelerate a Competitive Learning Algorithm Applied to VQ Codebook, *TEMA: Tendências em Matemática Aplicada e Computacional*, **11**:3 (2010), 193-203.

[6] C.Y. Chang, D.-F. Zhuang, A Fuzzy-Based Learning Vector Quantization Neural Network for Recurrent Nasal Papilloma Detection, *IEEE Trans. Circuits Syst. I*, **54**:12 (2007), 2619-2627.

[7] C. Chang et al., Reversible Information Hiding for VQ Indices Based on Locally Adaptive Coding, *Journal of Visual Communication and Image Representation*, **20**:1 (2009), 57-64.

[8] C.-C. Chang, C.-Y. Lin, Y.-P. Hsieh, Data Hiding for Vector Quantization Images using Mixed-Base Notation and Dissimilar Patterns without Loss of Fidelity, *Information Sciences*, (2012).

[9] W. Chen, M. Wang, A Fuzzy C-Means Clustering-Based Fragile Watermarking Scheme for Image Authentication, *Expert Systems with Applications*, **36**:2 (2009), 1300-1307.

[10] R.O. Duda, P.E. Hart, D.G. Stork, *Pattern Classification*, Wiley-Interscience, 2001.

[11] A.M. Filippi, J.R. Jensen, Effect of Continuum Removal on Hyperspectral Coastal Vegetation Classification using a Fuzzy Learning Vector Quantizer, *IEEE Trans. Geosci. Remote Sensing*, **45**:6 (2007), 1857-1869.

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

[13] N. Gkalelis, A. Tefas, I. Pitas, Combining Fuzzy Vector Quantization with Linear Discriminant Analysis for Continuous Human Movement Recognition, *IEEE Trans. Circuits Syst. for Video Technol.*, **18**:11 (2008), 1511-1521.

[14] R.M. Gray, Y. Linde, Vector Quantizers and Predictive Quantizers for Gauss-Markov Sources, *IEEE Trans. Commun.*, **30**:2 (1982), 381-389.

[15] N.B. Karayiannis, P.-I. Pai, Fuzzy Vector Quantization Algorithms and Their Applications in Image Compression, *IEEE Trans. Image Processing*, **4**:9 (1995), 1193-1201.

[16] D. Lee, S. Baek, K. Sung, Modified K-Means Algorithm for Vector Quantizer Design, *IEEE Signal Processing Lett.*, **4**:1 (1997), 2-4.

[17] E.A. Lima, G.G.M. Melo, W.T.A. Lopes, M.S. Alencar, F. Madeiro, Um Novo Algoritmo para Atribuição de Índices: Avaliação em Quantização Vetorial de Imagem, *TEMA: Tendências em Matemática Aplicada e Computacional*, **10** (2009), 167-177.

[18] C. Lin, S. Chen, N. Hsueh, Adaptive Embedding Techniques for VQ-Compressed Images, *Information Sciences*, **179**:1-2 (2009), 140-149.

[19] Y. Linde, A. Buzo, R.M. Gray, An Algorithm for Vector Quantizer Design, *IEEE Trans. Commun.*, **28**:1 (1980), 84-95.

[20] F. Madeiro, W.T.A. Lopes, M.S. Alencar, B.G. Aguiar Neto, Construção de Dicionários Voltados para a Redução da Complexidade Computacional da Etapa de Codificação da Quantização Vetorial, em "Anais do VI Congresso Brasileiro de Redes Neurais (CBRN'03)", 439-444, São Paulo-SP, (2003).

[21] F. Madeiro, W.T.A. Lopes, Introdução à Compressão de Sinais, *Revista de Tecnologia de Informação e Comunicação*, **1** (2011), 33-40.

[22] K.K. Paliwal, V. Ramasubramanian, Comments on Modified K-Means Algorithm for Vector Quantizer Design, *IEEE Trans. Image Processing*, **9**:11 (2000), 1964-1967.

[23] S.-T. Pan, S.-F. Liang, T.-P. Hong, J.H. Zeng, Apply Fuzzy Vector Quantization to Improve the Observation-Based Discrete Hidden Markov Model-An Example on Electroencephalogram (EEG) Signal Recognition, em "Proc. of the 2011 IEEE Int. Conf. on Fuzzy Systems", (2011).

[24] T. Phanprasit, T. Leauhatong, C. Pintavirooj, M. Sangworasil, Image Coding using Vector Quantization Based on Wavelet Transform Fuzzy C-Means and Principle Component Analysis, em "Proc. of the 9th Int. Symp. on Commun. and Information Technol. (ISCIT'2009)", (2009).

[25] A. Srinivasan, Speaker Identification and Verification using Vector Quantization and Mel Frequency Cepstral Coefficients, *Engineering and Technology*, **4**:1 (2012), 33-40.

[26] G.E. Tsekouras, A Fuzzy Vector Quantization Approach to Image Compression, *Applied Math. and Computation*, **167**:1 (2005), 539-560.

[27] R.T.Vieira, N. Brunet, S.C. Costa, S. Correia, B.G. Aguiar Neto, J.M. Fechine, Combining Entropy Measurements and Cepstral Analysis for Pathological Voice Assessment, *Journal of Medical and Biological Engineering (Article in press)*, (2012).

[28] R. Xu, D. Wunsch, Survey of Clustering Algorithms, *IEEE Trans. Neural Networks*, **16**:3 (2005), 645-678.