QUANTIZAÇÃO VETORIAL ADAPTATIVA MULTIESCALAS COM OTIMIZAÇÃO TAXA-DISTORÇÃO

Murilo B. de Carvalho

Depto. de Eng. de Telecomunicações
Universidade Federal Fluminense
R. Passos da Pátria, 156
Niteroi - RJ, 24210-240, BRASIL
murilo@lps.ufrj.br

Eduardo A. B. da Silva

PEE/COPPE/DEL/EE
Universidade Federal do Rio de Janeiro
Cx. P. 68504,
Rio de Janeiro, RJ, 21945-970, BRASIL
eduardo@lps.ufrj.br

Weiler Alves Finamore

CETUC, PUC-Rio
Rio de Janeiro, RJ, BRASIL
weiler@cetuc.puc-rio.br

RESUMO

Apresentamos um novo algoritmo para quantização vetorial adaptativa otimizado segundo um critério de taxa-distorção. Ele se baseia no casamento aproximado de padrões recorrentes multi-escalas. Nesta abordagem, o vetor de entrada é segmentado em blocos de tamanho variável. Os blocos são codificados usando um conjunto de dicionários, um para cada tamanho de bloco. Os dicionários são atualizados enquanto o dado é codificado, sem a necessidade de nenhuma informação lateral. Também não é requerido nenhum treinamento prévio. Foram usadas técnicas de programação dinâmica para otimizar a árvore de segmentação. O algoritmo apresenta bom desempenho para uma vasta gama de fontes, com resultados muito bons para fontes altamente não estacionárias, como é o caso de documentos compostos.

1. INTRODUÇÃO

Em um trabalho recente [1], foi descrita uma nova classe de algoritmos universais de compressão com perdas de dados muiti-dimensionais, representada pelo algoritmo UMMP (Universal Multiscale Matching Pursuits). Este algoritmo emprega um dicionário de vetores de diferentes tamanhos e um procedimento recursivo de segmentação para codificar segmentos de tamanhos variáveis do vetor de entrada. Isso pode ser visto como um quantizador vetorial adaptativo de dimensão variável (VQ). Exemplos de trabalhos anteriores em quantização vetorial adaptativa incluem [2, 3, 4]. As características que diferenciam o UMMP destes métodos são a técnica de atualização do seu dicionário, que não requer nenhuma informação lateral, e sua abordagem multi-escalas. O algoritmo UMMP tenta codificar um vetor de entrada usando um vetor do dicionário . Se a distorção na aproximação é maior do que um certo limiar, o vetor de entrada é dividido em dois segmentos e o procedimento completo é recursivamente repetido, com cada novo segmento sendo interpretado como um novo vetor de entrada, até que a distorção caia abaixo do limiar. O dicionário é atualizado pela concatenação dos vetores codificados anteriormente, no espírito do algoritmo sem perdas de Lempel-Ziv (LZ)[5]. Diferentemente do LZ entretanto, a segmentação do UMMP pode ser facilmente estendida para o caso de fontes multidimensionais ao invés de vetores. Também, sempre que um novo vetor de comprimento N é obtido por concatenação, o UMMP faz uma predicção de quais vetores devem ser incluídos nos dicionários correspondentes a todas as escalas.

Um dos pontos fracos do UMMP é que a segmentação criada usando decisões baseadas no cálculo da distorção local é sub-otima. Neste trabalho, aplicam-se conceitos de taxa-distorção para otimizar a segmentação em um VQ adaptativo semelhante ao UMMP. Este quantizador vetorial adaptativo usa um conjunto de $K = \log_2(N)$ dicionários $\mathcal{D}^{(k)}$ para codificar o dado. Um vetor de entrada $\mathbf{X} = \begin{pmatrix} X_0 & \dots & X_{N-1} \end{pmatrix}$ é segmentado em L segmentos, $\mathbf{X} = \begin{pmatrix} \mathbf{X}^{l_0} & \dots & \mathbf{X}^{l_{L-1}} \end{pmatrix}$ de comprimento $\ell(\mathbf{X}^{l_m})$, $m = 0, 1, \dots, L-1$. Esta segmentação pode ser representada por uma árvore de segmentação binária $\mathcal{S}$ como na figura 1.

Um nó da árvore é denotado por n_l. Este nó pode ter

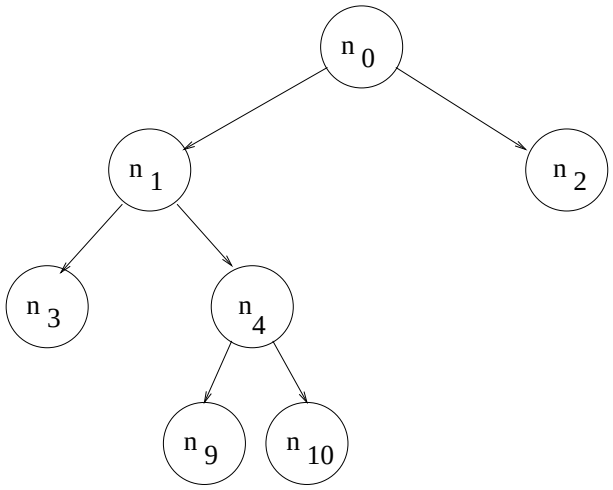


Figura 1: Uma árvore de segmentação binária.

dois nós filhos, n_{2l+1} e n_{2l+2}, ou nenhum filho. Um nó sem nenhum filho é uma folha. O nó raiz n_0 da árvore de segmentação corresponde a um segmento de comprimento N. Seus dois filhos, n_1 e n_2 são associados aos dois segmentos de comprimento $N/2$. Cada nó na profundidade p representa um segmento de comprimento $2^{-p}N$. Observa-se que a segmentação é dada pelas folhas, sendo o comprimento $\ell(\mathbf{X}^{l_m})$ do segmento $\mathbf{X}^{l_m}$ igual ao comprimento do nó da folha correspondente na árvore. Por exemplo, a segmentação representada pela árvore da figura 1 é $\mathbf{X} = \begin{pmatrix} \mathbf{X}^3 & \mathbf{X}^9 & \mathbf{X}^{10} & \mathbf{X}^2 \end{pmatrix}$ e os comprimentos dos segmentos são, respectivamente, $N/4, N/8, N/8$ e $N/2$.

Cada segmento $\mathbf{X}^{l_m}$ é codificado usando um elemento do dicionário correspondente $\mathcal{D}^{(k_m)} = \left\{\mathbf{S}_0^{(k_m)}, \ldots, \mathbf{S}_{M_{k_m}-1}^{(k_m)}\right\}$, onde $k_m = \log_2\left(\ell(\mathbf{X}^{l_m})\right)$. Isto é, o vetor de entrada é aproximado como $\hat{\mathbf{X}} = \begin{pmatrix} \mathbf{S}_{i_0}^{(k_0)} & \ldots & \mathbf{S}_{i_{L-1}}^{(k_{L-1})} \end{pmatrix}$. O algoritmo atualiza seus dicionários enquanto codifica o dado de entrada como se segue: Sempre que os segmentos correspondentes aos dois nós filhos do nó n_j forem codificados, o vetor resultante da sua concatenação $\hat{\mathbf{X}}^j = \begin{pmatrix} \hat{\mathbf{X}}^{2j+1} & \hat{\mathbf{X}}^{2j+2} \end{pmatrix}$ é incluído nos dicionários. Notar que esta atualização é feita no espírito do algoritmo de Lempel-Ziv sem perdas [5]. Para atualizar todos os dicionários, o comprimento deste vetor é modificado por uma transformação de escala $T_{2^k}^{\ell(\hat{\mathbf{X}}^j)}[\hat{\mathbf{X}}^j]$ para ajustar-se a cada dicionário $\mathcal{D}^{(k)}$. A transformação de escala é a função $T_N^M : \mathbb{R}^M \rightarrow \mathbb{R}^N$ que mapeia um vetor de comprimento M em um vetor de comprimento N. A saída do algoritimo é uma sequência de inteiros consistindo de indices do dicionário i_m e na seqüência de flags binários b_n que especificam a segmentação da árvore. Os flags representam $\mathcal{S}$ como uma série de decisões binárias, partindo-se da raiz para as folhas. Se, por exemplo, o flag binário 0 for usado para indicar segmentação e o flag 1 para indicar um nó folha, então a árvore da figura 1 seria representada pela sequência de flags 0,0,1,0,1,1,1.

2. A OTIMIZAÇÃO DA ÁRVORE DE SEGMENTAÇÃO

Cada nó folha n_l é associado a um segmento do vetor de entrada $\mathbf{X}^l$ que é representado por um elemento $\mathbf{S}_{i_l}^{(k_l)}$, onde $k_l = \log_2(\ell(\mathbf{X}^l))$. Assim sendo, pode-se avaliar a distorção associada ao nó n_l da seguinte forma :

$$\begin{aligned} D(n_l) &= \|\mathbf{X}^l - \mathbf{S}_{i_l}^{(k_l)}\|, \\ k_l &= \log_2(\ell(\mathbf{X}^l)) \end{aligned} \tag{1}$$

A taxa $R(n_l)$ é a taxa necessária para especificar o índice i_l, e é dada por:

$$R(n_l) = -\log_2(Pr(i_l|k_l)) \tag{2}$$

onde $Pr(i_l|k_l)$ é a probabilidade de ocorrência do índice i_l no dicionário da escala k_l.

A distorção total é:

$$D(\mathcal{S}) = \sum_{n_l \in \mathcal{S}_\mathcal{L}} D(n_l) \tag{3}$$

onde $\mathcal{S}_\mathcal{L}$ é o conjunto de nós folha de $\mathcal{S}$.

A quantidade de bits necessária para codificar esta aproximação é a taxa: $R(\mathcal{S})$, e é dada por:

$$R(\mathcal{S}) = R_t(\mathcal{S}) + \sum_{n_l \in \mathcal{S}_\mathcal{L}} R(n_l) \tag{4}$$

onde $R_t(\mathcal{S})$ é a taxa requerida para especificar a árvore de segmentção.

A melhor segmentação $\mathcal{S}^*$, no sentido taxa-distorção, leva à taxa mínima $R(\mathcal{S})$ dado que a distorção $D(\mathcal{S})$ não é maior que a distorção alvo D^* ou, alternativamente, leva à distorção mínima na taxa R^*. Este é um problema de minimização com restrições:

$$\begin{aligned} \mathcal{S}^* &= \arg\min_{\mathcal{S} \in \mathcal{S}_{R^*}} D(\mathcal{S}), \\ \mathcal{S}_{R^*} &= \{\mathcal{S} : R(\mathcal{S}) = R^*\} \end{aligned} \tag{5}$$

Para obter $\mathcal{S}^*$ pode-se encontrar a solução através do método dos multiplicadores de Lagrange λ. É sabido que se encontramos o mínimo do custo Lagrangeano $J(\mathcal{S}) = D(\mathcal{S}) + \lambda R(\mathcal{S})$, podemos também encontrar a solução para o problema de contorno quando escolhemos

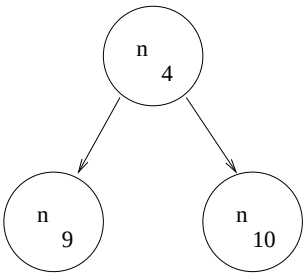


Figura 2: A sub-árvore $\mathcal{S}(n_4)$.

$R(\lambda) = R^*$ [6]. Ou seja:

$$\begin{aligned}\mathcal{S}^* &= \arg\min_{\mathcal{S}} J(\mathcal{S}) \\ &= \arg\min_{\mathcal{S}} \left(\sum_{n_l \in \mathcal{S}_{\mathcal{L}}} D(n_l) \right) + \lambda \left(R_t(\mathcal{S}) + \sum_{n_l \in \mathcal{S}_{\mathcal{L}}} R(n_l) \right) \\ &= \arg\min_{\mathcal{S}} \lambda R_t(\mathcal{S}) + \sum_{n_l \in \mathcal{S}_{\mathcal{L}}} (D(n_l) + \lambda R(n_l)) \\ &= \arg\min_{\mathcal{S}} \lambda R_t(\mathcal{S}) + \sum_{n_l \in \mathcal{S}_{\mathcal{L}}} J(n_l) \end{aligned} \quad (6)$$

onde $J(n_l) = D(n_l) + \lambda R(n_l)$.

Uma sub-árvore $\mathcal{S}(n_l)$ de $\mathcal{S}$ no nó n_l é a árvore binária com todos os nós de $\mathcal{S}$ tendo n_l como o nó raiz. A figura 2 ilustra a sub-árvore $\mathcal{S}(n_4)$ da árvore binária na figura 1. Denota-se $\mathcal{S} - \mathcal{S}(n_l)$ a árvore obtida de $\mathcal{S}$ pela podagem da sub-árvore $\mathcal{S}(n_l)$.

Se os custos Lagrangeanos $J(n_l)$, associados com a aproximação de cada segmento $\mathbf{X}^l$, são independentes, então o custo Lagrangeano de duas sub-árvores $J(\mathcal{S}(n_l))$ e $J(\mathcal{S}(n_m))$ são também independentes, desde que todos os nós de ambas sub-árvores sejam diferentes. Assim um algoritmo de busca rápido, similar a [4], pode ser implementado considerando-se que se $J(n_l) \leq J(\mathcal{S}(n_{2l+1})) + J(\mathcal{S}(n_{2l+2}))$ então as sub-árvores $\mathcal{S}(n_{2l+1})$ e $\mathcal{S}(n_{2l+2})$ devem ser podadas de $\mathcal{S}$ para diminuir o custo. Infelizmente este não é o caso do nosso VQ, porque os custos $J(n_l)$ são acoplados pelo processo de atualização do dicionário. Entretanto, se os dicionários iniciais são grandes o suficiente, a contribuição para a minimização de $J(n_l)$ devido à atualização do dicionário pode ser desprezada. Nas implementações VQ práticas, tende-se a usar um limite superior para o tamanho M do vetor de entrada $\mathbf{X}$ e para o número dos vetores nos dicionários a fim de lidar com a quantidade finita de memória disponível. Portanto, o dado de entrada é quebrado em blocos de tamanho M que são processados sequencialmente pelo VQ. Apesar de não ser verdadeiro para os primeiros blocos, os dicionários eventualmente crescem muito, o suficiente para que o custo Lagrangeano $J(n_l)$ possa ser quase desacoplado. Neste caso, poderia-se usar o algoritmo em [4] para obter uma solução aproximadamente ótima num sentido taxa-distorção.

Entretanto, se queremos usar tamanhos de blocos relativamente grandes ou o dicionário é muito pequeno (como acontece em taxas muito baixas), deve-se usar um algoritimo que leve em consideração o impacto do processo de atualização do dicionário. O dicionário é atualizado pela inclusão da concatenação de segmentos codificados previamente. Se escolhermos podar a sub-árvore, o impacto no custo não fica restrito a esta sub-árvore, mas pode afetar todos os nós que estão no lado direito da sub-árvore. Isto é, se podarmos uma sub-árvore na intenção de reduzir o custo, corremos o risco de remover do dicionário um elemento que poderia ser selecionado mais tarde como o melhor para aproximar um segmento de entrada. A ausência deste elemento provoca um aumento do custo. A ideia é podar uma sub-árvore somente quando o crescimento potencial no custo dos nós subsequentes, devido à remoção de alguns vetores do dicionário, não é maior que a redução no custo provida pela podagem. O Algoritmo é descrito abaixo:

passo 1 Inicialize $\mathcal{S}$ como a árvore completa de profundidade $\log_2(M) + 1$.

passo 2 Faça $J_i = \infty$ para os M nós folha, isto é, para $i = M-1, M, \ldots, 2M-2$.

passo 3 Faça $p = \log_2(M)$ e $\mathcal{S}_0 = \mathcal{S}$.

passo 4 Para cada nó $n_l \in \mathcal{S}$ na profundidade p, isto é, para $l \in \{2^{p-1}-1, 2^{p-1}, \ldots, 2^p - 2\}$, calcule:

(i) $J_l = J(n_l) + \lambda R_{1_l}$, onde $J(n_l)$ é o custo de representar o segmento de entrada associado ao nó n_l e R_{1_l} é a taxa necessária para indicar que o nó n_l é uma folha.

(ii) $\Delta J_l = \sum_{n_r \in \mathcal{S} - \mathcal{S}(n_l)} (J(n_r) - J'(n_r))$, onde $J'(n_l)$ é computado usando o dicionário sem $\hat{\mathbf{X}}^l = \begin{pmatrix} \hat{\mathbf{X}}^{2l+1} & \hat{\mathbf{X}}^{2l+2} \end{pmatrix}$, isto é, o dicionário que seria obtido sem a sub-árvore $\mathcal{S}(n_l)$.

passo 5 Se $J_l - J_{2l+1} - J_{2l+2} - \lambda R_{0_l} \leq \Delta J_l$ então pode os nós n_{2l+1} e n_{2l+2} de $\mathcal{S}$. (R_{0_l} é a taxa necessária para indicar a partição, e J_{2l+1}, J_{2l+2} foram computados na iteração anterior com $p+1$). Caso contrário, o custo do nó n_l é atualizado com $J_{2l+1} + J_{2l+2} + \lambda R_{0_l}$.

passo 6 Faça $p = p - 1$.

passo 7 Repita os passos 4 a 6 até que $p = 0$.

passo 8 Se $\mathcal{S} = \mathcal{S}_0$ então a otimização foi realizada. Caso contrário, vá ao passo 3.

É interessante considerar o porque de ΔJ_l ser avaliado para todos os nós, uma vez que apenas os nós folhas contribuem para o custo total. A ideia do algoritmo é podar apenas quando houver certeza que o custo não aumentará. O cálculo de ΔJ_l deve ser conservador porque não sabemos, no instante em que estamos decidindo sobre o nó n_l, quais nós serão folhas (as folhas atuais poderão ser podadas mais tarde). Quando avaliamos $\Delta J(n_l) > 0$, decidimos não podar. Entretanto, os nós que afetaram a decisão podem ser podados mais tarde. Deste modo, o procedimento completo deve ser repetido para melhorar a segmentação, até a convergência.

3. RESULTADOS EXPERIMENTAIS

O algoritmo para otimização taxa-distorção da árvore de segmentação descrito na seção anterior foi implementado e aplicado à compressão de imagens estáticas de níveis de cinza. A segmentação foi adaptada à característica bidimensional da fonte do seguinte modo: o nó n_0 corresponde a blocos 16×16 . Os nós na profundidade 1 correspondem a blocos de tamanho 8×16, os nós na profundidade 2 correspondem a blocos 8×8 e assim por diante. Os nós na profundidade p correspondem a blocos de tamanho $2^{\lfloor(\frac{8-p}{2})\rfloor} \times 2^{\lfloor(\frac{9-p}{2})\rfloor}$ ($\lfloor x \rfloor$ é o maior inteiro que é menor ou equal a x). A transformação de escala foi implementada usando o procedimento clássico de mudança de taxa de amostragem [8]. A seqüência inteira de índices do dicionário i_n foi codificada usando um codificador aritmético adaptativo com um modelo independente para cada escala. A seqüência de flags b_n foi codificada pelo codificador aritmético com diferentes modelos para cada profundidade. As taxas $R(n_l)$, R_{0_l} and R_{1_l} foram estimadas usando o logaritmo da frequência relativa de ocorrência dos simbolos usados pelos modelos do codificador aritmético.

O algoritmo foi aplicado à imagem Lena 512×512 e às imagens pp1209 e pp1205, ambas de dimensções 512×512, mostradas nas figuras 1a e 2a, respectivamente (devido a limitação de tamanho de arquivo apenas uma janela 400×400 delas foi exibida). Estas imagem foram obtidas por meio de captura por um scanner das páginas 1205 e 1209 da revista *IEEE Transactions on Image Processing*, Volume 9, número 7, Julho de 2000. A pp1209 é uma composição de imagens Lena comprimidas com texto e gráficos, enquanto a pp1205 possui somente texto. As figuras 3, 4 e 5 mostram a Relação Sinal-Ruído de Pico (PSNR) versus a taxa em bits/pixel obtida com estas imagens para o algoritmo original e para o algoritimo otimizado. As figuras 1b e 2b mostram as imagens reconstruídas usando o algoritmo proposto a 0.50 bits/pixel. A reprodução do texto e do gráfico é boa. Resultados para os algoritimos SPIHT [7] também são mostrados.

A melhoria do algoritmo otimizado R-D sobre o origi-

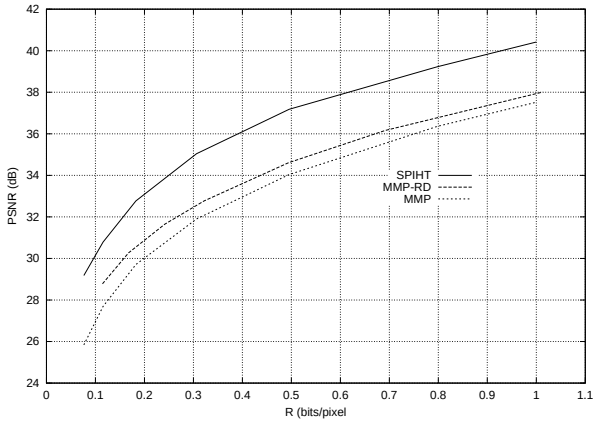


Figura 3: Desempenho para LENA 512×512.

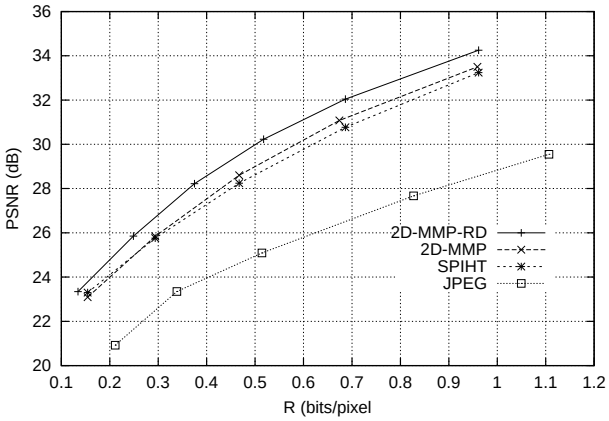


Figura 4: Desempenho para pp1209 512×512.

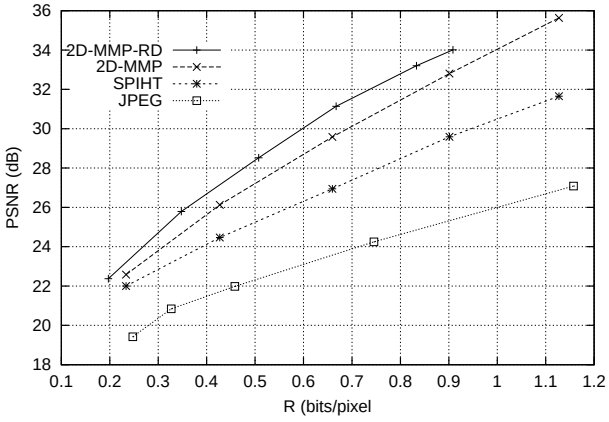
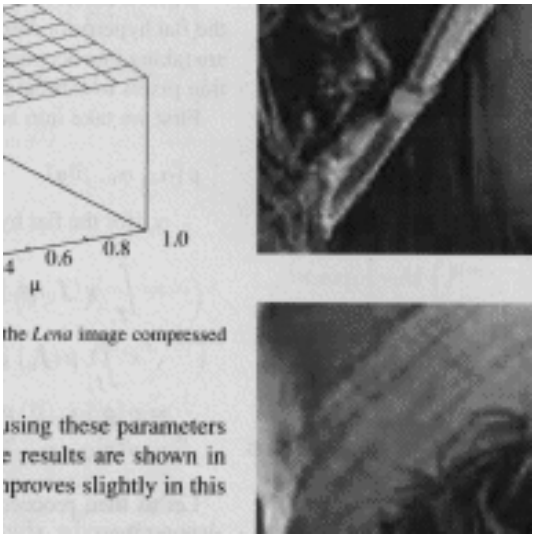
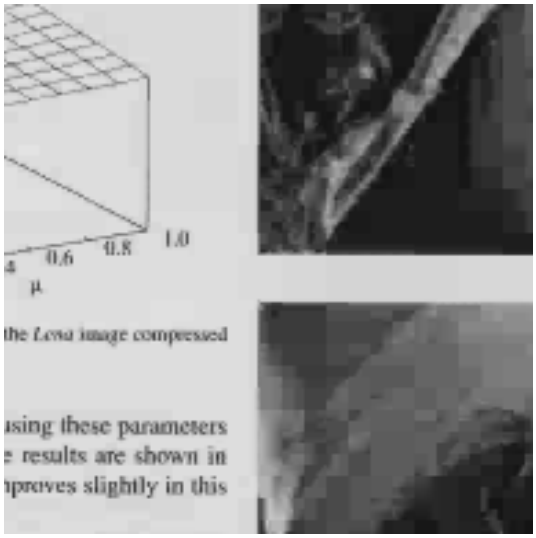


Figura 5: Desempenho para pp1205 512×512.

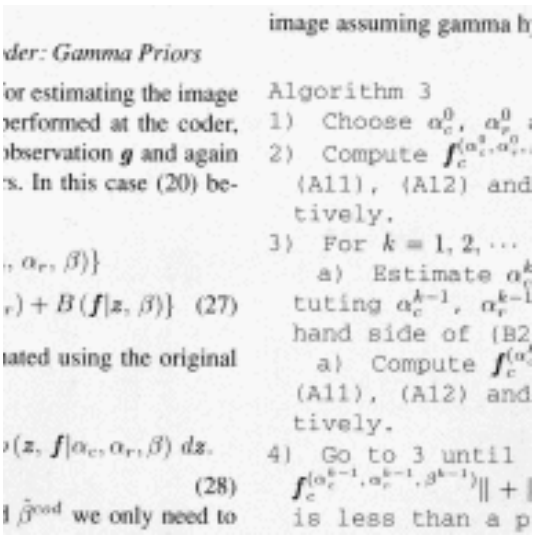


(a)

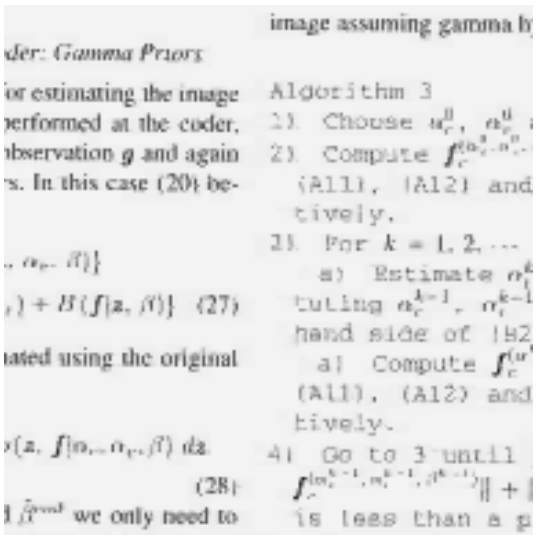


(b)

Tabela 1: Detalhe da imagem pp1209: (a) original; (b) comprimida a 0.50 bits/pixel.



(a)



(b)

Tabela 2: Detalhe da imagem pp1205: (a) original; (b) comprimida a 0.50 bits/pixel.

nal é clara, correspondendo a um ganho em torno de 1 dB em PSNR. O desempenho com a imagem Lena é 2 dB pior que o do algoritmo SPIHT. Entretanto o algoritmo proposto supera SPIHT em 1 dB com a imagem composta. Com documentos de apenas texto, o algoritimo supera o SPIHT em 5 dB.

4. CONCLUSÃO

Foi apresentado um novo algoritmo para quantização vetorial adaptativa. Ele é similar ao UMMP, um algoritimo universal para compressão com perdas apresentado anteriormente em [1], mas com uma árvore de segmentação otimizada. Diferente das abordagens clássicas do VQ, ele possui carater universal, pois ele constrói o dicionário enquanto codifica o dado de entrada, dispensando a necessidade de treinamento prévio do dicionário. A atualização do dicionário adota uma técnica tal que nenhuma informação lateral é necessária. O algoritimo segmenta o dado de entrada em blocos de tamanho variável. Ele utiliza múltiplos dicionários, um para cada comprimento de bloco. Seu desempenho é bastante promissor. Por exemplo, apesar de ser apenas um VQ aplicado diretamente na imagem, ele pode codificar documentos compostos superando um codificador baseado em " wavelet", o codificador SPIHT, por mais de 1 dB.

5. REFERÊNCIAS

[1] M. B. Carvalho and E. A. B. Silva, "A universal multi-dimensional lossy compression algorithm", *1999 IEEE International Conference on Image Processing*, October 1999, Kobe, Japan.

[2] M. Effros, P. A. Chou, and R. M. Gray, "One-pass adaptive universal vector quantization," *Procedings of ICASSP'94*, Vol. 5, pp. 625-628, Adelaide, 1994.

[3] C. Chan and M. Vetterli, "Lossy compression of individual signals based on string matching and one pass codebook design," *Procedings of ICASSP'95*, pp. 2491-2494, Detroit, 1995.

[4] G. J. Sullivan and R. L. Baker, "Efficient quadtree coding of images and video," *IEEE Transactions on Image Processing*, vol.3, No. 3, pp. 327-331, May 1994.

[5] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Transactions on Information Theory*, vol. it-24, No. 5, pp. 530-536, September 1978.

[6] R. E. Blahut, "Principles and Practice of Information Theory" Addison-Wesley publishing Company, 1988.

[7] A.Said and W.A. Pearlman, "A new, fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol.6, pp.243–250, June 1996.

[8] P. P. Vaidyanathan, "Multirate Systems and Filter Banks," Prentice-Hall Inc., 1993.