Aplicando o Algoritmo CTW no Padrão JPEG-LS

Diogo Chadud Milagres e Marcelo da Silva Pinho

***Resumo*— Este artigo apresenta um método para aplicar o conceito do algoritmo *context-tree-weighting* no padrão JPEG-LS. O objetivo desta adaptação é melhorar o desempenho do JPEG-LS quando este é utilizado para comprimir fontes com diferentes modelos probabilísticos. Enquanto o padrão utiliza uma forma específica para modelar a dependência estatística de pixels vizinhos, este trabalho propõe a utilização de um algoritmo para ponderar diferentes modelos. Resultados de simulação mostram que o JPEG-LS pode ser superado significativamente por outros algoritmos, quando ele é usado para comprimir imagens sintéticas, geradas a partir de modelos probabilísticos diferentes que o adotado pelo padrão. Além disso, é mostrado que o método proposto consegue atingir bons resultados para estas imagens sintéticas. O novo esquema é utilizado para comprimir um conjunto de imagens naturais e é mostrado que seu desempenho é similar ao desempenho do JPEG-LS. É ressaltado ainda que o modelo ponderado pode superar o JPEG-LS em algumas regiões de uma imagem natural. Um algoritmo capaz de calcular um limitante para o desempenho do esquema proposto também é apresentado. A diferença entre este limitante e o desempenho do padrão está em torno de 7% para o grupo de imagens naturais.**

***Palavras-Chave*— JPEG-LS, compressão de imagens, context-tree weighting, codificação universal**

***Abstract*— This paper proposes a new method to apply the concept of the context-tree-weighting algorithm in the JPEG-LS standard. The aim of this adaptation is to improve the performance of JPEG-LS when compressing sources with different models. While the standard works with a specific rule to model the statistical dependence of neighbor pixels, this work proposes the use of an algorithm to weight different models. Simulation results show that JPEG-LS can be outperformed significantly by different algorithms, when it is used to compress synthetic images, generated by using a model which is different than the model adopted in the standard. Furthermore, the proposed method achieves good results for these synthetic images. The new scheme is used to compress a group of natural images and it is shown that its performance is similar to the JPEG-LS performance. It is also pointed out that the weighted model can outperform the standard in some regions of a natural image. A procedure is presented to compute a bound to the performance of the proposed scheme. The difference between the standard and the bound is around 7%.**

***Keywords*— JPEG-LS, image compression, context-tree-weighting, universal coding**

I. INTRODUÇÃO

O padrão JPEG-LS foi proposto pelo comitê "ISO SC29/WG1" (JPEG) em 1999, com o objetivo de estabelecer um padrão para a compressão sem perdas ou quase sem perdas para imagens em tons de cinza e imagens coloridas [1]. A motivação de se criar um padrão deste tipo teve sua origem em aplicações onde o efeito da distorção causada por um sistema com perdas é crítico. Aplicações médicas envolvendo diagnósticos baseados em imagens são exemplos muito citados onde é desejável um sistema sem perdas ou quase sem perdas.

Tendo como base o algoritmo LOCO-I, o padrão JPEG-LS sugere um método de compressão de baixa complexidade computacional, baseado em um sistema de predição adaptativa seguido de um código de entropia [2]. Na verdade, uma transformação inversível seguida de um código de entropia é um procedimento tipicamente utilizado em sistemas para a compressão sem perdas, vide os algoritmos CALIC [3], SPIHT sem perdas [4] e JPEG2000 sem perdas [5]. O objetivo da transformação inicial é reduzir a dependência estatística entre pixels vizinhos para facilitar o projeto do código de entropia.

Conforme apontado em [2], um esquema comum de compressão de dados envolve dois pontos fundamentais: a escolha de um modelo probabilístico para a fonte de informação, e a escolha de um método de codificação. No desenvolvimento do padrão JPEG-LS, a imagem transformada é modelada como se fosse a saída de uma fonte com 729 estados diferentes, i.e., a matriz de pixels é modelada como uma matriz de variáveis aleatórias tal que a medida de probabilidade de um determinado pixel depende de quatro pixels vizinhos (que indicam em qual dos 729 diferentes estados a leitura da imagem se encontra quando o referido pixel é lido). O método de codificação utilizado é o código de Golomb, projetado para uma medida de probabilidade estimada a partir da imagem.

O modelo probabilístico utilizado no padrão JPEG-LS para a imagem transformada foi obtido após testes exaustivos, visando atingir boas taxas de compressão para diferentes grupos de imagens. No entanto, é óbvio que o modelo utilizado pelo padrão não é o melhor para qualquer fonte de informação. De fato, se a imagem transformada fosse gerada por uma fonte sem memória, gerando uma matriz de variáveis aleatórias estatisticamente independentes, a melhor solução seria considerar um modelo sem memória. O problema de se projetar um código para uma fonte cujo modelo probabilístico é desconhecido é estudado pela teoria da codificação universal [6].

A teoria da codificação universal visa projetar codificadores eficientes para qualquer fonte pertencente a uma determinada classe. Em 1995, Willems, Shtarkov e Tjalling mostraram que para fontes binárias com memória finita, a melhor solução possível pode ser atingida através da utilização de uma estrutura em árvore, denominada CTW (*context tree weighting*) [7].

Nesse trabalho, o conceito da CTW é utilizado para gerar diferentes versões do JPEG-LS. Esse artigo está organizado da seguinte forma. Na seção II, são mostradas as principais características do padrão JPEG-LS. O algoritmo CTW é apresentado na seção III. A contribuição deste trabalho, que é a aplicação dos conceitos da CTW no JPEG-LS é o assunto

Diogo Chadud Milagres e Marcelo da Silva Pinho, Divisão de Engenharia Eletrônica, Instituto Tecnológico de Aeronáutica, São José dos Campos, Brasil, E-mails: diogo@ele.ita.br, mpinho@ieee.org. Este trabalho foi financiado pela FAPESP (03/04607-5).

da seção IV. Os resultados são apresentados na seção V e fechando o trabalho, a seção VI apresenta as conclusões.

II. JPEG-LS

O padrão para a compressão de imagens sem perdas ou quase sem perdas do comitê JPEG, denominado JPEG-LS, é baseado no algoritmo LOCO-I, proposto em [8]. Conforme apontado na introdução o método de compressão se baseia em uma predição adaptativa seguida de um código de entropia.

A predição é o passo que visa reduzir a dependência estatística dos pixels, com o objetivo de facilitar o projeto de um código de entropia eficiente. Seja x o pixel que se deseja prever. Sejam a, b, c e d os pixels vizinhos ao x, conforme ilustra a Figura 1. A predição é feita em duas etapas. A primeira é obtida através de um processo não linear que não varia ao longo da imagem. Neste caso, o valor obtido pela predição é dado por

$$\hat{x}_{MED} = \begin{cases} \min(a,b), & \text{se } c \geq \max(a,b); \\ \max(a,b), & \text{se } c \leq \min(a,b); \\ a+b-c, & \text{caso contrário.} \end{cases} \quad (1)$$

Através de 1, é possível observar que quando $c \geq \max(a,b)$, o sistema supõe que existe uma borda vertical, se $a > b$, ou horizontal caso contrário. Sendo assim, $\hat{x}_{MED}$ assume o valor b se $a > b$ (borda vertical) ou assume o valor a se $b > a$ (borda horizontal). Analogamente, é possível observar que quando $c \leq \min(a,b)$, o sistema também opera supondo a existência de uma borda. Quando $\min(a,b) < c < \max(a,b)$ o sistema realiza uma predição linear, supondo que não está operando em uma região de borda. Por esta razão, este sistema também recebe o nome de detector de bordas.

c	b	d
a	x	

Fig. 1. Os pixels utilizados na predição do JPEG-LS

Em [9], foi observado que dado os pixels vizinhos a, b, c e d, o sistema de predição apresentado acima, em geral, produz um erro de predição, $x - \hat{x}_{MED}$ cuja média (ao longo da imagem) é diferente de zero. Na verdade, esta média do erro de predição é um viés existente no sistema. Para melhorar o desempenho do compressor, o JPEG-LS sugere uma correção adaptativa, que é a segunda etapa da predição. Ao invés de operar com os quatro pixels a, b, c e d, a predição se baseia nas diferenças (gradientes) $g_1 = d-b$, $g_2 = b-c$ e $g_3 = c-a$. Além disso, para evitar o problema da diluição de contextos [3], o JPEG-LS opera com versões quantizadas da tripla (g_1, g_2, g_3). Cada gradiente é quantizado através de um sistema com nove níveis de quantização, dando origem a versão quantizada (q_1, q_2, q_3). As nove regiões de quantização são $\{0\}$, $\{1,2\}$, $\{-2,-1\}$, $\{3,\ldots,6\}$, $\{-6,\ldots,-3\}$, $\{7,\ldots,20\}$, $\{-20,\ldots,-7\}$, $\{21,\ldots\}$ e $\{\ldots,-21\}$. Seja B a média do erro de predição ao longo dos pixels da imagem (que já foram lidos), cujos pixels vizinhos produzem uma dada tripla (q_1, q_2, q_3). Sendo assim, a segunda etapa refina a predição ajustando o valor para

$$\hat{x} = \hat{x}_{MED} + \lceil B \rceil \quad (2)$$

Após a predição, o padrão JPEG-LS utiliza um código de Golomb para codificar os erros de predição $e = x - \hat{x}$. O código de Golomb foi introduzido em [10] e é útil para codificar variáveis aleatórias com distribuição geométrica (OSGD - *one-sided geometric distribution*). Seja U uma variável aleatória com OSGD, i.e.,

$$p(u) = (1-q)q^u \text{ se } u \leq 0 \quad (3)$$

É possível mostrar que a entropia de U é dada por

$$H(U) = \frac{1}{1-q}\left\{-q\log_2(q) - (1-q)\log_2(1-q)\right\}. \quad (4)$$

É fácil verificar que quando $q = 2^{-1}$, $H(U) = 2$. Além disso, também é fácil mostrar que para este caso particular, existe um código ótimo fácil de ser implementado. De fato, basta mapear a realização u em uma palavra código com u bits iguais a zero seguido de um bit igual a um, indicando o fim da palavra código. Este código é conhecido como *código de vírgula*. Se $q = 2$, a média do comprimento da palavra código é igual a entropia.

Para projetar um código simples e eficiente para uma variável aleatória geométrica, U, com um valor de q arbitrário, o código de Golomb utiliza um inteiro ℓ para gerar duas variáveis aleatórias V e R, onde $V = \lfloor \frac{U}{\ell} \rfloor$ e $R = \frac{U}{\ell} - V$. Quando $\ell = \lceil \frac{q}{2(1-q)} \rceil$, é possível mostrar que V se aproxima de uma OSGD, tal que

$$p(v) \approx \frac{1}{2^{v-1}} \text{ se } v \geq 0,$$

e que R se aproxima de uma variável aleatória que assume valores no conjunto $\{0,\ldots,\ell-1\}$ com probabilidade $\frac{1}{\ell}$ (variável aleatória equiprovável). Como a transformação de U em V é inversível, então

$$H(U) = H(V) + H(R),$$

e sendo assim, um código eficiente para a dupla (V, R) produzirá um código eficiente para U. Como R se aproxima de uma variável aleatória equiprovável, um bom código é o de comprimento fixo. Além disso, foi visto acima que para uma OSGD com $q = 2^{-1}$, o *código de vírgula* é ótimo. Por esta razão, o código de Golomb utiliza estes dois métodos para codificar o par (V, R).

Para ilustrar o código de Golomb, seja U uma variável aleatória com distribuição geométrica, dada por (3), com $q = 0.84$, e que se deseja codificar a realização $u = 9$. Sendo assim, $\ell = 4$, $v = 2$ e $r = 1$ e a palavra código seria dada por 001 01.

Conforme apontado em [11], o código de Golomb também pode ser utilizado de forma eficiente para codificar variáveis aleatórias com distribuição geométrica bilateral (TSGD - *two-sided geometric distribution*).

Para codificar o erro de predição, o JPEG-LS considera que dada a tripla (q_1, q_2, q_3), o erro é bem modelado por uma variável aleatória com TSGD. No padrão, o parâmetro ℓ é estimado a partir dos pixels que já foram codificados, considerando a tripla (q_1, q_2, q_3), e para reduzir a complexidade computacional, o parâmetro m utilizado é uma potência de 2.

Através da descrição acima, é possível observar que o modelo utilizado pelo padrão JPEG-LS considera que o erro $x - \hat{x}_{MED}$ é bem modelado por uma TSGD com média diferente de zero, cujos parâmetros dependem do estado (q_1, q_2, q_3). Este modelo foi escolhido devido ao seu bom desempenho prático para diferentes grupos de imagens [2]. No entanto, da teoria da codificação universal, é possível afirmar que resultados melhores seriam obtidos se o modelo utilizado fosse otimizado para cada imagem, i.e., se o modelo da dependência estatística entre $x-\hat{x}_{MED}$ e os pixels vizinhos a x fosse otimizado para cada imagem.

III. O ALGORITMO CTW

O algoritmo CTW foi proposto em [7] com o objetivo de resolver o problema estudado pela teoria da codificação universal, quando a fonte de informação pertence a classe de fontes binárias com memória finita. Após a publicação do trabalho original, algumas extensões foram propostas para diferentes classes de fonte [12], [13].

Seja S uma fonte de informação cuja saída é uma seqüência de variáveis aleatórias, $X_1, \ldots, X_n, \ldots$, que podem assumir valores em um alfabeto finito. Uma fonte é de memória finita, se existe k tal que para todo i,

$$p(x_i|x_1, \ldots, x_{i-1}) = p(x_i|x_{i-k}, \ldots, x_{i-1})$$

A teoria da codificação universal tem como objetivo estudar métodos eficientes para se codificar a saída de uma fonte, quando sua medida de probabilidade é desconhecida ou parcialmente desconhecida. Se o modelo probabilístico da fonte de informação pode ser descrito através de uma máquina de estados, com um número finito de estados, e se esta máquina de estados é conhecida, exceto suas pelas medidas de probabilidade, é possível projetar um codificador eficiente para qualquer medida de probabilidade estacionária. De fato, utilizando o método proposto por Krichevsky e Trofimov em [14], conhecido como *estimador KT*, é possível estimar as medidas de probabilidade. Caso estas medidas sejam usadas para projetar um codificador aritmético, é possível mostrar que a taxa de bits obtida na codificação nos n primeiros símbolos é tal que

$$\rho_n = \frac{H(X_1^n)}{n} + O\left(\frac{\log n}{n}\right), \tag{5}$$

para qualquer fonte estacionária, onde $H(X_1^n)$ é a entropia de X_1^n [15].

Quando o modelo probabilístico é desconhecido mas se sabe que a fonte possui memória finita menor ou igual a k_m, uma possível solução para o problema da codificação universal é projetar os códigos baseados no *estimador KT* e no código aritmético para todos os modelos de fonte com memória menor ou igual a k_m e escolher o código que produz a menor taxa de bits. Neste caso, o modelo escolhido deve ser codificado e adicionado como um cabeçalho a palavra código. Em [15] é provado que para qualquer fonte markoviana estacionária, este método produz uma taxa de bits

$$\rho_n = \frac{H(X_1^n)}{n} + O\left(\frac{\log n}{n}\right). \tag{6}$$

Ainda em [15] é demonstrado que não existe um codificador capaz de produzir uma taxa menor que a apresentada em (6), que é denominada limitante de Rissanen.

Embora o sistema descrito acima atinja o limitante apresentado em (6), é possível notar que sua complexidade computacional é muito alta e desta forma, o sistema é pouco prático para fontes com memória. O algoritmo CTW é uma solução elegante para o problema da codificação universal para uma subclasse das fontes com memória finita e atinge o limitante de Rissanen [7]. De fato, usando uma estrutura de árvore, o algoritmo CTW estima uma medida de probabilidade, $\hat{p}(x_1^n)$, para a realização x_1^n, que corresponde a uma ponderação das medidas de probabilidade de todos os modelos possíveis, vide lema 2 em [7].

Da descrição do algoritmo CTW pode ser observado que a probabilidade usada para codificar o símbolo x_i, $i = 1, \ldots, n$ é uma média ponderada das probabilidades estimadas para contextos com diferentes comprimentos. Seja s^m a seqüência $x_{i-m}, \ldots, x_{i-1}$, $m = 0, \ldots, k_m$ (para $m = 0$, s^m é a seqüência de comprimento nulo) e k_m é a memória máxima da fonte. Seja $\hat{p}_{s^m}(x_i)$ a probabilidade estimada de X_i assumir o valor x_i, dado o contexto s^m. Sendo assim, a probabilidade utilizada pelo algoritmo CTW é dada por

$$p_{ctw}(x_i) = \sum_{m=0}^{k_m} \lambda_m \hat{p}_{s^m}(x_i), \tag{7}$$

onde $(\lambda_0, \ldots, \lambda_{k_m})$ depende da seqüência passada $x_1, \ldots, x_{i-1}$ e é tal que $\sum_{m=0}^{k_m} \lambda_m = 1$. Além disso, é importante citar que o cálculo dos pesos $(\lambda_0, \ldots, \lambda_{k_m})$ é realizado através da estrutura em árvore e devido a sua complexidade, será omitido neste trabalho. Mais detalhes podem ser encontrados em [7].

A partir de (7) é possível notar que a ponderação das medidas de probabilidade estimadas para diferentes contextos é a idéia central do algoritmo CTW. Portanto, é natural questionar qual o benefício de se utilizar esta técnica no padrão JPEG-LS.

IV. APLICANDO O CONCEITO DO ALGORITMO CTW NO PADRÃO JPEG-LS

Na seção II foi visto que o padrão JPEG-LS considera que o erro $x - \hat{x}_{MED}$ é bem modelado através de uma TSGD com média que pode ser diferente de zero (viés do detector de bordas). Os parâmetros da distribuição, i.e., a média e a taxa de decaimento, são considerados dependentes da tripla (q_1, q_2, q_3). Dentro do contexto da teoria da codificação universal, adotar este critério significa assumir que o modelo da fonte é conhecido. Embora este modelo apresente bons resultados para diferentes imagens, dificilmente este modelo é o melhor para todas as imagens. Sendo assim, a solução proposta pelo padrão poderia ser refinada considerando apenas que a fonte possui memória limitada. Neste caso, seria importante adotar algum método que atinja um bom desempenho para qualquer fonte dentro da classe de fontes com memória finita. Devido aos resultados da teoria da codificação de fonte, apresentados na seção III, o candidato natural seria o algoritmo CTW.

Partindo da idéia central do algoritmo CTW, resumida através de (7), é possível alterar o padrão JPEG-LS de tal forma que o erro $x - \hat{x}_{MED}$ seja codificado utilizando uma ponderação de modelos baseados em contextos de diferentes comprimentos. Como o que define o contexto no padrão é a tripla (q_1, q_2, q_3), a adaptação proposta neste trabalho utiliza os seguintes contextos diferentes, s^0 é a ausência de contexto, $s^1 = q_3$, $s^2 = (q_2, q_3)$ e $s^3 = (q_1, q_2, q_3)$. Com base nos diferentes contextos, s^m, $m = 0, \ldots, 3$, são calculadas as médias ao longo das imagens, que são representadas neste trabalho por B^m. A segunda etapa da predição, que é a correção adaptativa, é modificada para contemplar os diferentes modelos. No método proposto, o refinamento da predição é realizado como se segue.

$$\hat{x} = \hat{x}_{MED} + \lceil \sum_{m=0}^{3} \lambda_m B^m \rceil, \tag{8}$$

onde λ_m são pesos que podem ser ajustados para otimizar a codificação. Uma opção de ajuste para os pesos é a utilização da *context tree weighting*.

Da mesma forma que a média B^m depende do contexto, a taxa de decaimento das TSGD's dependem de s^m e conseqüentemente o parâmetro ℓ varia de acordo com o contexto selecionado. Para cada contexto s^m, o método proposto calcula um parâmetro ℓ^m e o código de Golomb projetado utiliza um parâmetro ponderado, conforme apresentado a seguir.

$$\ell_w = \sum_{m=0}^{3} \lambda_m \ell^m \tag{9}$$

É interessante observar que o método proposto não é simplesmente uma aplicação direta do algoritmo CTW como código de entropia de um compressor de imagens existente. De fato, a proposta deste trabalho é utilizar o conceito deste codificador universal para melhorar o modelo probabilístico adotado no padrão JPEG-LS. Além disso, é interessante ressaltar que esta idéia pode ser estendida sem dificuldade para ser aplicada em outros padrões de compressão.

V. RESULTADOS

Com o objetivo de avaliar a ineficiência do padrão JPEG-LS na compressão de dados gerados por fontes com modelos diferentes do adotado no padrão, foram criadas imagens sintéticas cujos erros, $x - \hat{x}_{MED}$, foram gerados a partir de seqüências pseudo aleatórias independentes, com distribuição geométrica bilateral. Além do algoritmo JPEG-LS, foi implementado uma versão do JPEG-LS que opera de forma equivalente ao padrão, estimando o viés da predição e o parâmetro do código de Golomb, de forma independente dos gradientes (q_1, q_2, q_3). É possível observar que esta versão, denominada JPEG-LS de ordem 0, é a solução adequada para a codificação quando não há dependência estatística entre o erro e os gradientes.

O método proposto também foi implementado, e os pesos foram obtidos utilizando o algoritmo CTW. Como no caso das imagens, o alfabeto do erro não é binário, as probabilidades calculadas na estrutura em árvore decaem rapidamente e em poucos passos, mesmo utilizando variáveis com dupla precisão, podem existir valores que são aproximados por zero. Para resolver este problema, foram adotados dois critérios diferentes. O primeiro, reinicializa a árvore sempre que a probabilidade conjunta estimada para um grupo de pixels atinge um valor mínimo. Neste artigo, o algoritmo que utiliza esta estratégia é representado por JPEG-LS/CTW0 e nos testes, o valor mínimo utilizado foi 10^{-20}. O outro mecanismo adotado para contornar o problema de arredondamento foi a adoção de uma probabilidade mínima, i.e., se em algum ponto da árvore, existir uma probabilidade estimada menor que um determinado limiar, esta probabilidade assume o valor mínimo. Esta versão é denominada JPEG-LS/CTW1 e o valor mínimo utilizado nos testes foi 10^{-6}.

Foram geradas imagens sintéticas de diferentes tamanhos e com diferentes decaimentos referentes à TSGD. Todas as imagens geradas, foram simuladas com viés igual a zero. Os tamanhos das imagens sintéticas variaram de 32×32 até 128×128 e os decaimentos utilizados foram $0,3$; $0,5$ e $0,8$. Para cada tamanho e decaimento foram geradas 10 imagens e cada algoritmo foi aplicado em todas as imagens. As médias das taxas de bits obtidas são apresentadas na Tabela I. Como já era esperado, os resultados do JPEG-LS de ordem 0 são os melhores, pois as imagens foram geradas com base no modelo utilizado por este algoritmo. É interessante observar que para imagens com tamanho reduzido, o custo de se utilizar o JPEG-LS com modelo errado pode chegar a 14% (caso 32×32 e decaimento igual a $0,3$). Além disso, a utilização do conceito do algoritmo CTW é capaz de reduzir este custo para 2%.

TABELA I
TAXAS DOS CÓDIGOS (EM BITS POR PIXEL - BPP) PARA IMAGENS SINTÉTICAS.

Parâmetros		JPEG-LS			
Tamanho	decaimento	Padrão	ordem 0	CTW0	CTW1
32×32	0,3	2,47	2,12	2,24	2,18
	0,5	3,18	3,00	3,11	3,04
	0,8	4,91	4,72	4,77	4,74
64×64	0,3	2,36	2,10	2,16	2,13
	0,5	3,11	3,01	3,06	3,02
	0,8	4,83	4,71	4,75	4,72
128×128	0,3	2,30	2,09	2,12	2,11
	0,5	3,03	3,00	3,01	3,00
	0,8	4,75	4,71	4,72	4,71

Outro fato importante de ser observado nos resultados apresentados na Tabela I é que a medida que o tamanho da imagem cresce, a diferença de desempenho dos algoritmos diminui. De fato, quando o tamanho da imagem é 128×128 e o decaimento é igual a $0,8$, o custo de se utilizar o padrão é menor que 1%. Este fato também já era esperado, visto que quando o tamanho da imagem cresce indefinidamente, as taxas de bits convergem para o mesmo valor. Na verdade, a desvantagem do JPEG-LS está na velocidade da convergência, visto que este algoritmo não utiliza o modelo correto.

Como na maioria das aplicações, as imagens possuem tamanho bem maior que 128×128, pode parecer que a ineficiência do padrão para imagens geradas a partir de modelos diferentes é pouco relevante. No entanto, é importante lembrar que em geral, um processo estacionário nem sempre é bom para modelar imagens reais. No caso de processos localmente

estacionários, o custo de se utilizar o modelo errado pode se manter alto, mesmo para imagens maiores, visto que a convergência deve ocorrer em um pequeno pedaço da imagem. Para avaliar este efeito, foram geradas imagens de tamanho 512×512, com taxa de decaimento da TSGD variável ao longo da imagem. Para simular um processo localmente estacionário, o decaimento foi mantido constante em trechos da imagem. A Tabela II apresenta os resultados obtidos com decaimentos no intervalo $[0,3; 0,9]$, sendo que o decaimento é mantido constante durante K pixels. A partir destes resultados, é possível notar que a ineficiência do padrão é da ordem de 5%, para este caso.

TABELA II
TAXAS DOS CÓDIGOS (EM BPP) PARA IMAGENS SINTÉTICAS NÃO ESTACIONÁRIAS.

K	JPEG-LS			
	Padrão	ordem 0	CTW0	CTW1
512	3,87	3,65	3,72	3,76
1024	3,83	3,60	3,68	3,67
2048	3,84	3,61	3,69	3,66

Para avaliar o desempenho do algoritmo proposto em imagens naturais, foi utilizado um conjunto com sete imagens diferentes, todas elas de tamanho 512×512 e com 8 bits por pixel. As imagens testadas foram as seguintes: *airplane, baboon, barbara, lena512, peppers, sailboat* e *tiffany*, todas disponíveis na página na Internet do *Center for Image Processing Research* do *Rensselaer Polytechnic Institute* (http://www.cipr.rpi.edu). A Tabela III apresenta os resultados obtidos para as imagens naturais.

TABELA III
TAXAS DOS CÓDIGOS (EM BPP) PARA IMAGENS NATURAIS.

Imagens	JPEG-LS			
	Padrão	ordem 0	CTW0	CTW1
Airplane	3,64	3,97	3,63	3,83
Baboon	5,83	5,97	5,82	5,96
Barbara	4,92	5,35	4,92	5,15
Lena512	4,28	4,54	4,28	4,41
Peppers	4,40	4,78	4,42	4,60
Sailboat	4,84	5,16	4,85	5,06
Tiffany	3,94	4,27	3,94	4,10
Média	**4,55**	**4,86**	**4,55**	**4,73**

Os resultados da Tabela III mostram que para as imagens naturais utilizadas, o modelo utilizado pelo padrão produz melhores resultados que o modelo de ordem 0. Além disso, os resultados dos algoritmos que utilizam a ponderação, em geral, estão entre o padrão e o modelo de ordem 0. Estes resultados indicam que se a opção por um modelo fixo for feita, esta opção deve ser pelo modelo proposto no padrão. No entanto, existem casos (as imagens *baboon* e *airplane*) em que o algoritmo CTW0 produziu algum ganho em relação ao modelo do padrão, mesmo tendo este modelo um desempenho bem superior ao modelo de ordem zero. Este fato é um indício de que em uma mesma imagem, devido a não estacionariedade, a utilização de modelos diferentes ao longo da imagem pode trazer vantagens em relação ao uso do modelo fixo. Para avaliar esta questão, foi implementado um algoritmo que utiliza o melhor modelo para cada pixel. Como o modelo escolhido depende do erro de predição, este algoritmo não é um codificador válido, pois para recuperar o erro de predição, seria necessário conhecer o erro antes de decodificá-lo, o que obviamente não é possível. No entanto, este algoritmo mostra um limitante inferior para a ponderação. Os resultados obtidos são apresentados na Tabela IV e mostram uma redução de aproximadamente 7% em relação ao padrão.

TABELA IV
LIMITANTE DE DESEMPENHO PARA A PONDERAÇÃO (EM BPP) - IMAGENS NATURAIS.

Imagens	Limitante
Airplane	3,31
Baboon	5,53
Barbara	4,50
Lena512	4,00
Peppers	4,15
Sailboat	4,58
Tiffany	3,62
Média	**4,24**

Foi realizada uma análise no algoritmo que calcula o limitante para identificar as regiões onde cada um dos modelos foi escolhido. Foi observado que, em geral, os modelos com memória menor que três são escolhidos nas regiões de borda das imagens. Este fato está de acordo com a suposição de que a imagem é localmente estacionária. No entanto, como a seqüência de leitura da imagem não garante que dois pixels com estatísticas semelhantes sejam lidos um seguido do outro (p.ex. se a imagem é lida linha a linha, dois pixels de uma borda vertical serão lidos com o espaçamento de pelo menos w pixels, onde w é o número de colunas da imagem), a convergência do algoritmo CTW fica prejudicada. Os autores deste trabalho acreditam que seja possível encontrar algum mecanismo que seja capaz de ponderar os modelos de forma mais eficiente, analisando a região da imagem em que o algoritmo está operando. Neste caso, os algoritmos seriam capazes de se aproximar dos limitantes apresentados na Tabela IV.

VI. CONCLUSÕES

Este artigo apresentou uma forma de adaptar o conceito do algoritmo CTW no padrão JPEG-LS. Seguindo as recomendações da teoria da codificação universal, esta adaptação buscou projetar sistemas eficientes para diferentes modelos de fonte. Conforme pode ser observado na descrição do padrão, o JPEG-LS supõe que o erro de predição gerado pelo detector de bordas é bem modelado por uma variável aleatória com distribuição geométrica bi-lateral (TSGD), cuja média e decaimento dependem de quatro pixels vizinhos. Embora o método proposto também opere considerando uma TSGD, ele utiliza uma ponderação de modelos probabilísticos com diferentes regras para a dependência estatística entre os pixels. Para avaliar o custo de se utilizar o modelo errado na compressão, foram geradas imagens sintéticas, cujos erros de predição eram independentes dos pixels vizinhos. Os

resultados apresentados mostram que o JPEG-LS pode ter um desempenho 14% inferior a um algoritmo que utiliza o modelo correto. Além disso, foi visto que o método proposto possui desempenho apenas 2% pior do esquema que utiliza o modelo correto. Como as imagens naturais, em geral, não são bem modeladas por processos estacionários, este trabalho também avaliou o desempenho do método para imagens sintéticas geradas a partir de modelos localmente estacionários. Também neste caso, foi observado que o JPEG-LS é ineficiente quando aplicado em imagens geradas com modelos cujos erros de predição são independentes dos pixels vizinhos. O método proposto também se mostrou eficiente para as imagens sintéticas não estacionárias. O método proposto também foi aplicado em um grupo de imagens naturais. Os resultados obtidos mostraram que o método possui desempenho similar ao desempenho do JPEG-LS. A partir de resultados obtidos para algumas imagens, foi observado que pode existir vantagem em se utilizar diferentes modelos ao longo de uma imagem. Por esta razão, este trabalho calculou que resultado seria obtido caso fosse possível escolher o modelo ótimo para cada pixel. Na média, este sistema produz uma taxa em torno de 7% melhor que a taxa do JPEG-LS. Analisando as regiões onde os diferentes modelos foram utilizados, é possível notar que os modelos de menor ordem, em geral, são ótimos nas bordas existentes na imagem. Sendo assim, os autores deste trabalho acreditam que seja possível adaptar o método proposto para chegar próximo dos resultados obtidos com o algoritmo que utiliza o modelo ótimo.

REFERÊNCIAS

[1] ISO/IEC 14495-1, ITU-T Recommendation T.87. *Information technology lossless, and near-lossless compression of continuous-tone still images,* 1999.

[2] M. J. Weinberger, G. Seroussi, e G. Sapiro, "The loco-i lossless image compression algorithm: principles and standardization into jpeg-ls", *IEEE Transactions on Image Processing*, vol. 9, no. 8, pp. 1309-1324, 2000.

[3] X. Wu, e N. D. Memon, "Context-based adaptive lossless image coding", *IEEE Transactions on Communications*, vol. 45, no. 4, pp. 437-444, 1997.

[4] A. Said, e W. A. Pearlman, "An image multiresolution representation for lossless and lossy compression," *IEEE Transactions on Image Processing*, vol. 5, no. 9, pp. 1303-1310, 1996.

[5] D. S. Taubman, e M. W. Marcelin, *JPEG2000: Image Compression Fundamentals, Standards and Practice*, Kluwer Academic Publishers, Boston, 2001.

[6] J. Rissanen, e G. G. Langdon Jr., "Universal modeling an coding," *IEEE Transactions on Information Theory*, vol. 27, no. 1, pp. 12-23, 1981.

[7] F. M. J. Willems, Y. M. Shtarkov, e T. J. Tjalkens, "The context-tree weighting method: basic properties," *IEEE Transactions on Information Theory,* vol. 41, no. 3, pp. 653-664, 1995.

[8] M. J. Weinberger, G. Seroussi, e G. Sapiro, "LOCO-I: A low complexity, context-based, lossless image compression algorithm," *Proc. Data Compress Conference,* Snowbird, pp. 140-149, 1996.

[9] G. G. Langdon Jr. e M. Manohar, "Centering of context-dependent components of prediction error distributions," *Proceedings of SPIE (Applications of Digital Image Processing XVI)*, vol. 2028, Jul., pp. 26-31, 1993.

[10] S. W. Golomb, "Run-length encoding," *IEEE Transactions on Information Theory*, vol. 12, no. 3, pp. 399-401, 1966.

[11] N. Merhav, G. Seroussi, e M. Weinberger, "Optimal prefix codes for sources with two-sided geometric distributions," *IEEE Transactions on Information Theory*,vol. 46, no. 1, pp. 121-135, 2000.

[12] F. M. J. Willems, Y. M. Shtarkov, e T. J. Tjalkens, "Context weighting for general finite context sources," *IEEE Transactions on Information Theory*, vol. 42, no. 5, pp. 1514-1520, 1996.

[13] F. M. J.Willems, "The context-tree weighting method: extensions," *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 792-799, 1998.

[14] R. E. Krichevsky e V. K. Trofimov, "The performance of universal encoding," *IEEE Transactions on Information Theory*, vol. 27, no. 2, pp. 199-207, 1981.

[15] J. Rissanen, "Complexity of strings in the class of markov sources," *IEEE Transactions on Information Theory*, vol. 32, no. 4, pp. 526-532, 1986.