Codificação Fractal de Imagens

Ana Lúcia Mendes Cruz
UNICAMP
analucia@decom.fee.unicamp.br

Fernando Silvestre da Silva
UNICAMP
silva@decom.fee.unicamp.br

Yuzo Iano
UNICAMP
yuzo@decom.fee.unicamp.br

Roger Fredy Larico Chavez
UNICAMP
rlarico@decom.fee.unicamp.br

***Abstract*— This paper presents fractal bases and its potentiality when applied to image coding as a new ally on the establishment of new coding standards, especially when merged to the new techniques presented in literature. The basic topics about fractal compression, specially the PIFS coders (*Partitioned Iterated Function Systems*) are included, illustrative and mathematically. A really complete approach concerning the traditional literature and state of art literature follows about the topic including hybrid coders, such as a quick comparison involving fractal, wavelets, hybrids techniques and JPEG2000.**

***Index Terms*— Image Coding, Fractals, coding time, wavelets.**

***Resumo*— Este artigo tem por objetivo apresentar as bases e potencialidades da codificação fractal de imagens como aliada no estabelecimento de novos padrões de compressão, especialmente quando combinados a técnicas recentemente apresentadas na literatura. Os princípios básicos da compressão fractal, em especial os codificadores PIFS (*Sistema de Funções Iterativas Particionadas*), são apresentados, ilustrativa e matematicamente. Uma abordagem bastante completa acerca da bibliografia tradicional e estado da arte contendo inclusive algoritmos híbridos é apresenta da sobre o tema, culminando com uma breve comparação sobre sistemas fractais, wavelets, híbridos e JPEG2000.**

***Palavras chave*— Codificação de Imagens, fractais, velocidade de codificação, wavelets.**

I. INTRODUÇÃO

Imagens digitais exigem uma parcela cada vez maior do mundo da informação. Basta observar os avanços contínuos na tecnologia de impressoras, celulares e câmeras digitais na última década, por exemplo. Novas técnicas não somente transformaram as aplicações e serviços existentes, como a distribuição de vídeo para entretenimento residencial, mas também geraram novas indústrias e serviços tais como vídeo-conferência, distribuição por satélite direto às residências, gravação de vídeo digital, serviços vídeo *on demand*, HDTV (*High Definition Television*), vídeo em dispositivos móveis, *streaming* de vídeo, entre outros [1] [2].

Embora os métodos de compressão de imagens sejam de uso extensivo hoje, a demanda pelo aumento da capacidade de armazenamento e de velocidade de transmissão exige pesquisas contínuas em busca de novos métodos ou na melhoria dos já existentes. Muitas áreas da ciência, em especial a engenharia e matemática, têm se dedicado a essa questão. Novas tecnologias, a exemplo das técnicas fractais, se tornam aliadas potenciais no estabelecimento de novos padrões de compressão de imagens.

A. Codificação Fractal de Imagens

A codificação fractal de imagens consiste em representar os blocos da imagem através de coeficientes de transformações contrativas, explorando o conceito de auto-similaridade. Assim, nesse tipo de codificação, ao invés de armazenar/transmitir os blocos da imagem como uma coleção de *pixels*, somente são enviados/armazenados os coeficientes dessas transformações. Esse princípio de funcionamento permite obter altas taxas de compressão mantendo ótima qualidade visual.

Entre as vantagens da compressão fractal estão a rápida decodificação, transmissão progressiva, boa compressão e o fato de praticamente não necessitar de codificadores entrópicos [1][3][4]. Por se tratar de uma técnica ainda pouco explorada, se apresenta como um vasto campo para a pesquisa, dado o seu alto potencial para a compressão de imagens. Um dos aspectos que encorajam a codificação fractal é que ela é completamente paralelizável. Hardware paralelo especializado pode, portanto, reduzir significativamente o tempo de codificação.

Apesar das características vantajosas da compressão fractal, o tempo exaustivo de processamento é a principal barreira na implementação de sistemas práticos que façam uso dessa tecnologia [5][6]. Diversas tentativas recentes têm sido feitas na tentativa de minimizar esta questão, abordando desde vetores de características até redes neurais. Muita pesquisa ainda se faz necessária para destravar o potencial fractal latente para a compressão de imagens. Sem dúvida, essas pesquisas devem residir na questão de promover a aceleração

· Manuscrito recebido em 24 de março de 2007; revisado em 07 de junho de 2007.

A. L. M. Cruz (analucia@decom.fee.unicamp.br), F. S. Silva (silva@decom.fee.unicamp.br), Y. Iano (yuzo@decom.fee.unicamp.br), e R.Chavez (rlarico@decom.fee.unicamp.br) pertencem ao Departamento de Comunicações – DECOM da Faculdade de Engenharia Elétrica e de Computação da UNICAMP .

da etapa de compressão dessa nova tecnologia.

II. Sistema de Funções Iterativas (IFS): bases da compressão fractal de imagens

O termo fractal foi primeiramente introduzido por Benoit Mandelbrot em 1983 [7]. A propriedade-chave que caracteriza os fractais e os diferencia das demais técnicas é a auto-similaridade, isto é, os fractais apresentam a mesma complexidade de detalhamento independente da escala em que são observados. Barnsley and Sloan [8] foram os primeiros a reconhecer o potencial da aplicação da teoria fractal ao problema da compressão de imagens e patentearam sua idéia em 1990 e 1991 [9][10]. Era o chamado IFS (*Iterated Funcion Systems* ou Sistema de Funções Iterativas).

Em 92, Jacquin [11] introduziu o conceito de particionamento por blocos, criando a técnica conhecida como PIFS (*Partitioned Iterated Function System* ou Sistema de Funções Iterativas Particionadas). O PIFS é o sistema que melhor se adequa às imagens reais, sendo efetivamente utilizado nos codificadores fractais atuais [5]. Contudo, a compreensão do IFS é essencial para se entender o modo como a compressão fractal trabalha [12].

Uma analogia simples para explicar esses princípios básicos de funcionamento da idéia fractal foi feita por Fisher [3]: essa analogia consiste num tipo especial de fotocopiadora que reduz a imagem a ser copiada pela metade e a reproduz três vezes na cópia de saída (reduzidas e deslocadas), como mostrado na Fig. 1.

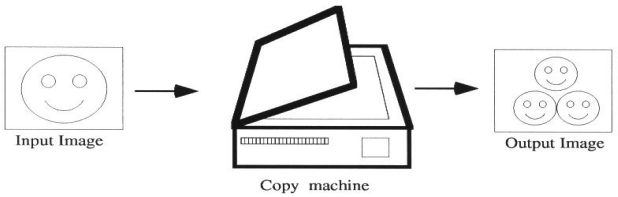


Fig. 1 – Exemplo de Fotocopiadora de Fisher. [3]

A idéia básica consiste em passar essa cópia de saída novamente pela copiadora, num processo recursivo. Após várias iterações desse processo, percebe-se na Fig. 2 que mesmo com diferentes imagens de entrada, todas as cópias de saída parecem convergir para a mesma imagem final, mostrada na Fig. 2c. Essa imagem é o chamado "atrator" para essa máquina fotocopiadora especifica. Logo, a imagem inicial não afeta o atrator final; de fato somente a posição e orientação das cópias (transformações afins) determinam como a imagem final se parecerá.

Uma vez que a fotocopiadora reduz a imagem de entrada, qualquer imagem inicial será reduzida a um ponto na medida em que aumentam as iterações. As transformações possuem, portanto, a limitação de serem "contrativas", ou seja, a distância entre dois pontos quaisquer da imagem de saída (após transformação) precisa ser menor do que a distância entre os pontos correspondentes na imagem de entrada [3].

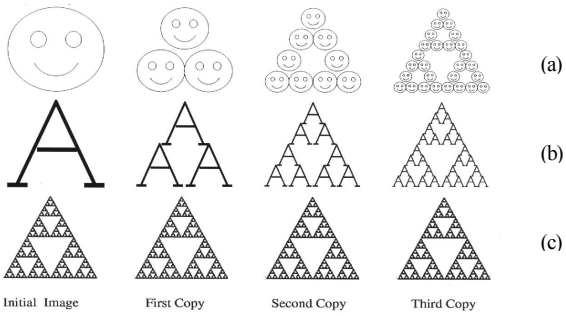


Fig. 2 – Primeiras 3 cópias geradas pela fotocopiadora para 3 imagens diferentes. [3]

Na prática, transformações na forma (1) permitem gerar uma grande variedade de interessantes atratores. Essas transformações afins polinomiais de primeira ordem podem torcer, deslizar, rotacionar, escalar ou deslocar a imagem de entrada, dependendo dos valores dos coeficientes.

$$\omega_i \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \end{bmatrix} \quad (1)$$

Assim, matematicamente, um sistema de funções iterativas (IFS) em R^2 consiste de uma coleção de transformações contrativas $\{\omega_i : R^2 \rightarrow R^2 \mid i=1,...n\}$ que mapeiam o plano R^2 para si mesmo em cada iteração. Essa coleção de transformações define o mapa [3]:

$$W(\cdot) = \bigcup_{i=1}^{n} \omega_i(\cdot) \quad (2)$$

Foi Hutchinson [13] quem provou que se algumas condições forem satisfeitas e se esse mapa W for contrativo, a recursão converge para o atrator. Logo, como o mapa W (ou os ω_i) determina completamente uma única imagem [3], é possível determinar o atrator, que é a imagem que deseja-se recuperar.

Assim, dada uma imagem inicial A_0, aplicando uma vez o conjunto de transformações W, gera-se $A_1=W(A_0)$; a segunda vez gerando $A_2=W(A_1)=W(W(A_0))\equiv W^{o2}(A_0)$ e assim por diante. Dessa forma, o atrator é o conjunto limite:

$$A_f = \lim_{n\to\infty} \mathrm{W}^{\mathrm{on}}(\mathrm{A}_0) \quad (3)$$

que independe da escolha de A_0.

Na Fig. 3 é apresentado um exemplo prático da aplicação desta teoria. Note que partiremos de uma mesma imagem e aplicaremos a ela três mapas contrativos diferentes. O primeiro mapa consiste em aplicar três transformações que simplesmente subamostram e deslocam a imagem. O segundo consiste em subamostragem com duas rotações mais inversão e duas subamostragens com dois deslocamentos. O terceiro mapa abrange rotações, subamostragens, contrações e deslocamentos. Note que dependendo do mapa aplicado na recursão, é gerado um atrator diferente. Note também que o atrator independe da imagem inicial. Trocar a imagem inicial por outra e aplicar as mesmas transformações recursivamente, fará o sistema convergir para o mesmo atrator.

Torna-se prioritário frisar que cada atrator é formado por copias reduzidas e transformadas de si mesmo, implicando na auto-similaridade. O atrator, ou seja, a imagem que deseja-se

recuperar é formado por auto-similaridades de si próprio. Este conceito é fundamental para se compreender os sistemas de codificação fractal utilizados hoje.

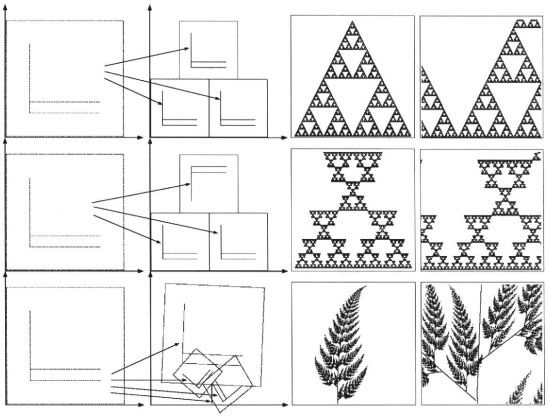


Fig. 3 – Transformações Afins, Atratores e Ampliação no Atrator. [3]

Finalmente, Barnsley [14] foi o primeiro a perceber que armazenar os coeficientes das transformações afins ao invés de armazenar ou transmitir a imagem como uma coleção de *pixels*, poderia gerar significativa compressão. Uma vez armazenados os coeficientes, para reconstruir a imagem basta aplicar recursivamente as transformações por eles determinadas em uma imagem qualquer.

Abordagens matemáticas mais detalhadas podem ser encontradas em [1][3][4][15][16], mas basicamente os dois principais teoremas nos quais esta fundamentada a teoria fractal apresentada são o Teorema do Mapeamento Contrativo e o Teorema da Colagem. As principais definições e propriedades acerca do Sistema de Funções Iterativas são apresentados a seguir.

A. Espaço em que os fractais existem e a Contratividade: abordagem matemática

Lema A (Contratividade): "Seja (X, d) um espaço métrico completo, onde d é a métrica associada que mede a distância entre os elementos do espaço X. Então, o espaço dos fractais ($\mathcal{H}(\boldsymbol{X})$, h) é o espaço dos subconjuntos compactos e não-vazios do espaço X, sendo também um espaço métrico completo [16]. Seja também $\{\omega_n : X \to X$, $n=1, 2, ..., N\}$ um conjunto de mapeamentos contrativos neste espaço, e portanto, em ($\mathcal{H}(\boldsymbol{X})$, h) (onde h é a métrica de *Hausdorff*) [16]. O fator de contratividade de ω_n é dado por s_n para cada n. Definindo $W : \mathcal{H}(\boldsymbol{X}) \to \mathcal{H}(\boldsymbol{X})$ por:

$$W(A) = \omega_1(A) \bigcup \omega_2(A) \bigcup \ldots \bigcup \omega_N(A) \tag{4}$$

$$W(A) = \bigcup_{n=1}^{N} \omega_n(A), \text{ para cada } A \in \mathcal{H}(\mathbf{X}). \tag{5}$$

então W é um mapeamento contrativo com fator de contratividade $s = max \{s_n , n=1, 2, ..., N \}$".

Através do lema A, os sistemas de funções iterativas são definidos por [16]:

Definição 1: "Um Sistema de Funções Iterativas – IFS – consiste em um espaço métrico completo ($\boldsymbol{X}$,d) e um conjunto finito de mapeamentos contrativos $\omega_n : \boldsymbol{X} \to \boldsymbol{X}$, com os respectivos fatores de contratividade s_n, para $n=1, 2, ..., N$. A notação para a IFS é $\{(\boldsymbol{X}, d)$; ω_n, $n=1, 2, ..., N\}$ e seu fator de contratividade é $s = max \{s_n , n=1, 2, ..., N \}$".

Teorema A: "Seja $\{(\boldsymbol{X}, d)$; ω_n, $n=1, 2, ..., N\}$ um sistema de funções iterativas com fator de contratividade s. Então a transformação $W : \mathcal{H}(\boldsymbol{X}) \to \mathcal{H}(\boldsymbol{X})$ definida por:

$$W(A) = \bigcup_{n=1}^{N} \omega_n(A) \tag{6}$$

para todo $A \in \mathcal{H}(\boldsymbol{X})$, é um mapeamento contrativo no espaço métrico completo ($\mathcal{H}(\boldsymbol{X})$, h) com fator de contratividade s, ou seja:

$$h(W(A), W(B)) \le s \cdot h(A, B), \tag{7}$$

para todo $A,B \in \mathcal{H}(\boldsymbol{X})$. A existência e unicidade do ponto fixo, $A_f \in \mathcal{H}(\boldsymbol{X})$, chamado de *atrator* da IFS, é dado pela aplicação do teorema do ponto fixo de mapas contrativos, que leva a:

$$A_f = \lim_{n \to \infty} W^{on}(A), \quad \forall A \in \mathcal{H}(\boldsymbol{X}), \tag{8}$$

de forma que $A_f = W(A_f)$.

A distância entre um dado $A \in \mathcal{H}(\boldsymbol{X})$ e o atrator A_f obedece à inequação

(TEOREMA DA COLAGEM):

$$h(A, A_f) \le \frac{1}{1-s} h(A, W(A))" \tag{9}$$

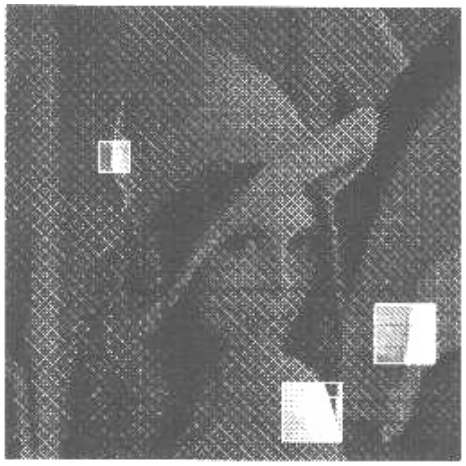
Esse teorema resume as características que possibilitam o modelamento de fractais como sendo subconjuntos compactos de um espaço métrico completo que pode ser completamente especificado através de um sistema de equações (mapas contrativos).

III. SISTEMA DE FUNÇÕES ITERATIVAS PARTICIONADAS (PIFS): CODIFICADORES FRACTAIS ATUAIS

Uma imagem real, em geral, não apresenta a auto-similaridade apresentada, dita "auto-similaridade global", ou seja, a imagem não parece conter transformações afins de si mesma. Ao invés disso, algumas de suas áreas apresentam similaridades entre si (similar por partes). Na Fig. 4 podem ser vistas similaridades entre as regiões de diferentes tamanhos destacadas no chapéu e espelho e entre regiões destacadas no ombro [3]. Para codificar essas imagens reais é necessária a aplicação do PIFS.

A distinção entre esse tipo de similaridade e a dos fractais apresentados na Fig. 3 é que, naquele caso, a imagem era formada por cópias transformadas de si mesma *inteira* e aqui a imagem é formada por cópias propriamente transformadas de *partes* de si mesma. Dessa forma, as partes transformadas geralmente não se encaixam perfeitamente. Por conseqüência, a imagem codificada pelo PIFS não será uma cópia idêntica da imagem original, mas sim uma aproximação [1].

(a)



(b)

Fig. 4 – (a) Imagem Lena Original (256 x 256 pixels); (b) Exemplos de similaridades nessa imagem. [3]

A. Codificação PIFS

Assim, a codificação PIFS se resume a procurar, para cada bloco da imagem que se deseja codificar, pelo bloco de tamanho maior que seja auto-similar. O bloco que se deseja codificar é chamado de *range-block*. Os blocos de tamanho maior que serão comparados com cada *range-block* são chamados de *domain-blocks*, que conjuntamente compõem o que se chama de *domain pool*, na verdade um espaço de procura que pode ou não conter informação parcialmente redundante (*overlap* entre *domain-blocks*). Assim, a cada *domain-block*, são aplicadas subamostragem, filtragem de média e transformações-afins ω_i (estas últimas compostas por uma parte geométrica e ajustes de contraste e brilho). Os resultados de todas essas transformações são comparados com o *range-block* que está sendo codificado [1].

A escolha pelo melhor *matching* (casamento) é feita minimizando uma medida de distorção. Ao encontrar o par *domain-range* somente são enviados os coeficientes da transformação afim que transforma o *domain-block* D_i no *range-block* R_i e o endereço do *domain-block* que foi escolhido como seu equivalente.

A necessidade de se encontrar o melhor *matching* surge uma vez que, em geral, a intenção de determinar a região D_i da imagem que após a transformação gere exatamente R_i não pode ser alcançada. Isso ocorre porque uma imagem real dificilmente é composta somente por partes que, após a transformação, se encaixem perfeitamente em outra parte da mesma imagem.

Assim, o que sempre é possível, é tentar encontrar uma outra imagem *f'* tal que $f' = f_\infty$ e que a métrica *d(f, f')* seja pequena o suficiente, ou seja, procura-se o mapa *W* tal que o seu ponto fixo *f'* seja próximo (ou se pareça com) a imagem *f*. A métrica *d(f, f')* pode ser definida de várias formas, contudo, a métrica utilizada na quase totalidade dos trabalhos em codificação fractal é a métrica *rms*, mostrada na equação (10), escolha que facilitará a determinação dos valores dos fatores de contraste s_i e brilho o_i, conforme será apresentado à frente.

$$d_{rms}(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\langle \boldsymbol{x}-\boldsymbol{y}, \boldsymbol{x}-\boldsymbol{y} \rangle} \tag{10}$$

Onde o operador <·,·> simboliza o produto interno padrão.

O processo de codificação, então, se resume a encontrar as regiões D_i e os mapas ω_i tais que ao aplicar ω_i em D_i obtém-se algo mais próximo possível da região R_i.

Para formalizar matematicamente a codificação de imagens com PIFS é necessário primeiramente definir o modelo matemático que será usado para a imagem natural:

Definição 2: "Seja $I = [0;1] \subset \mathfrak{R}$ o intervalo unitário na reta real e, conseqüentemente, I^2 o quadrado unitário no plano euclidiano. Uma imagem natural pode ser vista como o gráfico de uma função $f \in \boldsymbol{F}$ tal que $\boldsymbol{F} = \{ f \mid f : I^2 \rightarrow \mathfrak{R} \}$".

Deve-se notar que a função *f* mapeia o quadrado unitário para o intervalo unitário, mas considera-se que a imagem de *f* seja $\mathfrak{R}$ para que as somas e subtrações de imagens fiquem bem definidas. Embora o *range* e o *domain* da função sejam, respectivamente, R_i x *I* e D_i x *I*, é usual referenciá-los apenas como *range-block* R_i e *domain-block* D_i.

Suponha que se deseja codificar por PIFS uma imagem *f*. Ou seja, deseja-se encontrar a coleção de mapas $\omega_1, \omega_2, ..., \omega_n$ com:

$$\omega_i \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix}, \; i=1, 2, ..., n \tag{11}$$

onde $\{a_i, b_i, c_i, d_i, e_i, f_i\}$ define a transformação geométrica; s_i determina o contraste e o_i o brilho. Todos esses componentes compõem a transformação afim, mas por vezes, a transformação geométrica é referenciada separadamente, uma vez que é transmitida separadamente do brilho e contraste no *bitstream* do sinal codificado. A transformação geométrica é enviada em 3 bits. Tipicamente o brilho é enviado em 7 bits, e o contraste em 5 bits [3].

$W(f)$ é definido por:

$$W(f) = \bigcup_{i=1}^{N} \omega_i \left(f \cap \left(D_i \times I \right) \right) \tag{12}$$

de modo que $f = f_\infty$ seja o atrator de *W*. Em outras palavras, o

PIFS consiste na aplicação recursiva de um mapa W composto por uma coleção de transformações contrativas ω_i, com a diferença de que agora a imagem será processada por blocos, ou seja, cada transformação será aplicada somente sobre o *domain-block*, ao invés de ser aplicado na imagem toda. A união de todas as partes transformadas irá formar a nova imagem após a iteração.

Usando a notação $\omega_i(f)$ para indicar $\omega_i(f \cap (D_i \times I))$, a equação do ponto fixo pode ser expressa como:

$$f_\infty = W(f_\infty) = \omega_1(f_\infty) \cup \omega_2(f_\infty) \cup \ldots \omega_n(f_\infty). \quad (13)$$

O objetivo, na prática, é encontrar $f' = f_\infty$ tal que o erro $d_{sup}(f, f')$ seja pequeno o suficiente, conforme já colocado. Nesse caso:

$$f \approx f' = W(f') \approx W(f) = \omega_1(f) \cup \omega_2(f) \cup \ldots \omega_n(f) \quad (14)$$

Então, é suficiente aproximar os blocos-R_i dos blocos-D_i transformados minimizando a medida de distorção:

$$\min_{\omega_i} \{d\ (f \cap (R_i \times I), \omega_i(f))\}, \quad i = 1, \ldots, n \quad (15)$$

Quando se utiliza partição *quadtree* para processamentos da imagem, o formato de D_i e R_i é sempre quadrado e o tamanho do *domain-block* é, em geral, o dobro do tamanho do *range-block*. Assim, a escolha da parte geométrica do mapa é limitada a apenas uma redução de tamanho por um fator de 2 seguida de uma dentre as 8 opções: 4 rotações (0º, 90º, 180º e 270º) e 4 inversões (vertical, horizontal, diagonal principal, diagonal secundária). Assim, a minimização da equação (15) fica bastante simplificada, bastando encontrar os valores s_i e o_i que minimizam a distância em cada uma das 8 transformações geométricas possíveis.

B. Cálculo dos Valores de s_i e o_i

Sejam $\boldsymbol{r}_i$ o *range-block* transformado em vetor; $\boldsymbol{d}_i$ o *domain-block* transformado em vetor (após subamostragem e transformação geométrica); $\boldsymbol{1}$ o vetor de uns"; e *NxN* as dimensões do *range-block*. Para calcular os valores de s_i e o_i ótimos para cada candidato a *domain-block-equivalente* em uma dada transformação geométrica utiliza-se a expressão:

$$\min_{s_i\, o_i}\left(d_{rms}\left(\boldsymbol{r}_i,\ s_i \cdot \boldsymbol{d}_i + o_i \cdot \boldsymbol{1}\right)\right) = \left(\sum_{j=1}^{N^2} \left| r_{ij} - \left(s_i \cdot d_{ij} + o_i\right)\right|^2\right)^{1/2} \quad (16)$$

Para resolver essa equação utilizar-se-á uma a métrica *rms* mostrada na equação (10). Assim:

$$d^2_{rms}\left(\boldsymbol{r}_i,\ s_i \cdot \boldsymbol{d}_i + o_i \cdot \boldsymbol{1}\right) = \left\langle \boldsymbol{r}_i - \left(s_i \cdot \boldsymbol{d}_i + o_i \cdot \boldsymbol{1}\right), \boldsymbol{r}_i - \left(s_i \cdot \boldsymbol{d}_i + o_i \cdot \boldsymbol{1}\right)\right\rangle =$$
$$= s_i^2 \langle \boldsymbol{d}_i, \boldsymbol{d}_i \rangle + 2 o_i s_i \langle \boldsymbol{d}_i, \boldsymbol{1} \rangle + o_i^2 \langle \boldsymbol{1}, \boldsymbol{1} \rangle - 2 s_i \langle \boldsymbol{d}_i, \boldsymbol{r}_i \rangle - 2 o_i \langle \boldsymbol{1}, \boldsymbol{r}_i \rangle + \langle \boldsymbol{r}_i, \boldsymbol{r}_i \rangle \quad (17)$$

Diferenciando com relação a s_i e a o_i e igualando a zero obtém-se os valores:

$$s_i = \frac{\langle \boldsymbol{d}_i, \boldsymbol{1} \rangle \langle \boldsymbol{1}, \boldsymbol{r}_i \rangle - \langle \boldsymbol{1}, \boldsymbol{1} \rangle \langle \boldsymbol{d}_i, \boldsymbol{r}_i \rangle}{\langle \boldsymbol{d}_i, \boldsymbol{1} \rangle^2 - \langle \boldsymbol{1}, \boldsymbol{1} \rangle \langle \boldsymbol{d}_i, \boldsymbol{d}_i \rangle} \quad e \quad o_i = \frac{\langle \boldsymbol{d}_i, \boldsymbol{1} \rangle \langle \boldsymbol{d}_i, \boldsymbol{r}_i \rangle - \langle \boldsymbol{1}, \boldsymbol{r}_i \rangle \langle \boldsymbol{d}_i, \boldsymbol{d}_i \rangle}{\langle \boldsymbol{d}_i, \boldsymbol{1} \rangle^2 - \langle \boldsymbol{1}, \boldsymbol{1} \rangle \langle \boldsymbol{d}_i, \boldsymbol{d}_i \rangle} \quad (18)$$

Substituindo pela definição do produto interno:

$$s_i = \frac{\sum \boldsymbol{d}_i \cdot \sum \boldsymbol{r}_i - N^2 \cdot \sum \boldsymbol{d}_i \boldsymbol{r}_i}{\left(\sum \boldsymbol{d}_i\right)^2 - N^2 \cdot \sum \boldsymbol{d}_i^2} \quad e \quad o_i = \frac{\sum \boldsymbol{d}_i \cdot \sum \boldsymbol{d}_i \boldsymbol{r}_i - \sum \boldsymbol{r}_i \cdot \sum \boldsymbol{d}_i^2}{\left(\sum \boldsymbol{d}_i\right)^2 - N^2 \cdot \sum \boldsymbol{d}_i^2} \quad (19)$$

onde:

$\sum \boldsymbol{r}_i$ = soma de todos os elementos do vetor $\boldsymbol{r}_i$;

$\sum \boldsymbol{d}_i$ = soma de todos os elementos do vetor $\boldsymbol{d}_i$;

$\sum \boldsymbol{r}_i \boldsymbol{d}_i$ = soma do produto elemento a elemento (produto interno) dos vetores $\boldsymbol{d}_i$ e $\boldsymbol{r}_i$;

$\sum \boldsymbol{d}_i^2$ = soma do quadrado de todos os elementos do vetor $\boldsymbol{d}_i$;

N^2 = número de elementos de cada vetor.

Note que os valores de s_i e o_i calculados por estas equações não são limitados em termos de amplitude e precisarão ser quantizados. Para calcular o erro de aproximação usam-se os valores quantizados, pois serão estes os valores enviados ao decodificador. Outro fator de restrição é o máximo valor de contraste permitido s_{max} (que é um parâmetro do codificador). Caso o valor calculado ultrapasse s_{max}, o valor s_{max} é usado para o cálculo do erro [3].

C. Decodificação PIFS

Uma vez transmitidos os dados, cada iteração do processo de *decodificação* de imagens naturais em escala de cinza através do PIFS é realizado da seguinte forma: uma máscara seleciona uma parte de uma imagem no decodificador (*domain-block*), na qual aplica-se uma subamostragem e uma transformação-afim ω_i (já transmitida). A seguir, esse *domain-block*, já subamostrado e transformado, é copiado para uma região específica desta mesma imagem (*range-block*). Esse processo é repetido para todas as partes da imagem. O conjunto final de todos os *range-blocks* formará a imagem já decodificada após esta iteração [1].

Assim, para realizar a codificação e decodificação através do PIFS é necessária primeiramente a definição de quatro aspectos básicos [16]:

- O número de cópias transformadas que irão compor a imagem de saída, ou seja, o número de *range-blocks*. Em geral essa informação é transmitida de forma implícita, como no caso da partição *quadtree*;
- A máscara que selecionará qual a parte afetada (*domain-block*) em cada transformação.
- O ajuste de brilho e contraste para cada *domain-block*;
- Os demais coeficientes das transformações afins (além de contraste e brilho) associados a cada *domain-block*.

Dada uma imagem *f* em escala de cinza, uma iteração do processo de decodificação utilizando uma partição com *N*

partes (*N range-blocks*) pode ser descrito como a aplicação do mapeamento *W* da seguinte forma [16]:

$$W(f) = \omega_1(f) \cup \omega_2(f) \cup ..... \cup \omega_N(f)$$

onde ω_i é aplicada *somente* sobre a respectiva região D_i. É importante salientar que associado a cada ω_i tem-se uma região D_i na imagem de entrada e uma região R_i na imagem de saída.

Como *W(f)* representa uma imagem, deve-se salientar que $\bigcup R_i = I^2$ e que $R_i \bigcap R_j = \varnothing$, se $i \neq j$, ou seja, R_i é uma *partição* da imagem de saída (I^2). Dessa forma, pode-se dizer que o processo iterativo é tal que a saída da primeira iteração f_1 é dada por $f_1=W(f_0)$, a da segunda $f_2=W(f_1)=W(W(f_0))=W^{o2}(f_0)$ e assim por diante.

A Fig. 5 ilustra um exemplo do processo de decodificação. Parte-se de uma imagem qualquer, e aplica-se recursivamente o mapa. O resultado de algumas iterações pode ser observado. Em geral, após a quarta ou quinta iteração, a seqüência já converge.

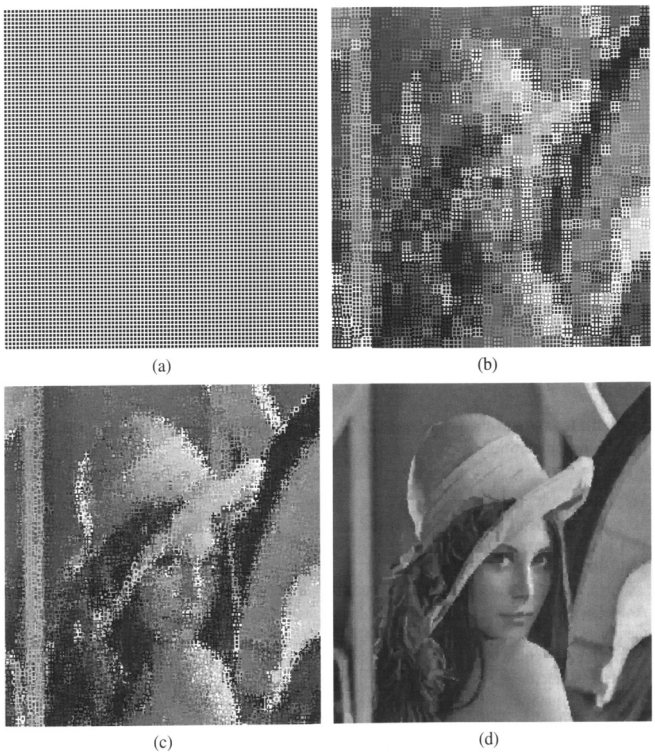


Fig.5 – Decodificação fractal da imagem Lena: a) imagem inicial arbitrária; b) após a 1ª iteração; c) após a 2ª iteração; d) após a 10ª iteração. [16]

IV. CODIFICAÇÃO FRACTAL ACELERADA

A codificação fractal apresenta ótima qualidade de imagens, mesmo a baixas taxas de bits e a decodificação fractal é extremamente rápida. Entretanto, na codificação, cada *range-block NxN* é comparado a *todos* os *domain-blocks 2Nx2N* da *domain pool* dentro de uma imagem *MxM*. Como conseqüência, o algoritmo de *matching* possui complexidade computacional $O[M^4]$, o que exige um tempo exaustivo de processamento. Esse peso computacional é o principal motivo de hesitação da adoção de aplicações fractais, sendo similar ao peso computacional do processo de estimação de movimento de sistemas de codificação de vídeo como, por exemplo, o MPEG2, onde o processo de procura pelo melhor *matching* é responsável em média por 40% do tempo de codificação.

Basicamente, as tentativas de acelerar a codificação fractal consistem em modificar os seguintes aspectos: a composição da *domain pool*, o tipo de procura usado no *block matching*, ou a representação/quantização dos parâmetros transformados.

Diferentes tentativas de reduzir o tempo de compressão fractal têm sido tentadas. As mais tradicionais dizem respeito à restrição da *Domain Pool* via pré-classificação dos blocos. Por exemplo, em [17] Jacquin classificou os blocos usando informações de borda e, em [3], Fisher pré-classificou os blocos segundo seus valores de média e variância. Fisher [3] desenvolveu uma classificação de domínios com o objetivo de reduzir o número de comparações de *matching*. O ponto principal desse esquema é que, primeiramente, todos os *domain-blocks* são classificados. Durante a codificação, cada *range-block* é classificado e somente será comparado com os *domain-blocks* de mesmo tipo que o seu (segundo um critério baseado em média e variância), o que reduz significativamente o número de comparações de *matching*.

Bogdan e Meadows [18] aplicaram uma rede auto-organizadora de Kohonen ao problema da codificação fractal, requerendo treinamento do codificador sobre a imagem a ser codificada. Esses autores reduziram o número de comparações *range-domain*, mas não a complexidade da comparação em si. McGregor *et. al.* [19], por outro lado, trabalharam esses dois problemas usando uma procura do tipo K-D-tree nos *domain-blocks* combinada à extração de um pequeno número de características do bloco-imagem.

Um passo importante na questão da aceleração fractal foi dado em [20], e foi a inspiração do codificador de Cardinal de [21]. Em [20] Saupe reduziu a questão da procura pelo melhor matching à procura pelo vizinho mais próximo dentro de um espaço métrico conveniente. Nesta técnica, cada bloco é associado a um vetor de características tal que minimizar o erro de colagem entre o *range-block* e o *domain-block* equivale a minimizar a distância entre os vetores de característica correspondentes. A procura pelo melhor *matching* então consiste em simplesmente encontrar o vetor mais próximo no espaço vetorial. Deve-se colocar, entretanto, que o método de Saupe encontra o melhor matching e só depois calcula os valores de *si* e *oi*, o que pode ferir a restrição de contratividade sobre *si*. O método de Saupe pode ser feito de forma mais rápida se for usada uma estrutura K-D-*tree*.

Em [21], Cardinal reduziu o processo de procura pelas transformações afins de cada *range-block* ao problema geométrico clássico de busca pelos vizinhos mais próximos num espaço euclidiano. Baseando-se na partição geométrica do espaço de características dos *range-blocks*, Cardinal conseguiu boa aceleração na compressão sem perda de

qualidade. Idéias similares são apresentadas em [22][23]. Também pode ser utilizada a quantização vetorial como em [24][25]. Em [4] um método ligeiramente menos sofisticado que o de Cardinal [21] é apresentado, mas segue a mesma filosofia. Cardinal argumenta que tais técnicas tendem a superar as aproximações clássicas que têm por base a redução da *Domain Pool*.

Em 2002, Tong e Wong [26] apresentaram uma procura pelo vizinho mais próximo baseada na projeção ortogonal e pré-quantização dos parâmetros fractais transformados. Tong e Pi também desenvolveram um codificador de procura adaptativa que exclui um grande número de *domain-blocks* não qualificados [27] e Wang e Hsieh [28] elaboraram um método que explora a correlação entre *range-blocks* com o mesmo propósito. Bani Eqbal [29] propôs também um método de procura em árvore para a *Domain Pool*.

Em 2002, Chi *et al.* [30] apresentaram um novo modelo de métrica baseado na integral *fuzzy* de Sugeno combinada com uma partição *quadtree* obtendo melhor qualidade visual e redução no tempo de compressão.

Alguns algoritmos eficientes [31][32][33] foram também apresentados para aliviar a carga computacional mantendo a mesma qualidade de imagem do *Full Search*. Entretanto, pré-processamento é necessário para esses algoritmos.

Wen *et.al.* também em 2003 [34] utilizaram média e variância como método de classificação combinado com técnicas de redução nas transformações. Também em 2003, Siu *et. al.* [35] propuseram um esquema baseado numa condição de exclusão única e numa predição de contraste zero. A condição de exclusão única evita um grande número de comparações de matching, enquanto a predição de contraste zero é capaz de determinar se o fator contraste para um domain-block é zero ou não, calculando também a diferença entre o *range-block* e o *domain-block* transformado de maneira eficiente e exata.

Huang *et. al.* [36] propuseram um algoritmo também baseado em variância introduzindo dois parâmetros capazes de controlar a velocidade de codificação e a qualidade e em [37] é proposto um método de compressão fractal de imagens usando um vetor de características especial com o objetivo de classificar os *domain-blocks* da imagem num algoritmo bastante conveniente para a implementação em hardware.

Ao lado dessas tentativas de acelerar a codificação fractal pura, alguns codificadores híbridos têm também sido desenvolvidos. A relação entre fractal e codificadores baseados em transformada foi investigada em [8]-[42]. Davis [40][43][44] apresentou uma aproximação que usa elementos de ambos os tipos de compressão: fractal e *wavelet*. Em [41] argumenta-se que o mapeamento contrativo fractal poderia ser considerado como uma operação de predição do domínio *wavelet* e em [42] os autores fizeram o matching *domain-range* no domínio *wavelet* para obter uma compressão mais alta.

Li e Kuo [45] usaram o mapeamento contrativo fractal para predizer os coeficientes *wavelets* entre escalas e então codificaram o resíduo de predição com um codificador de plano de bits. Esse procedimento é diferente de [39] e [40] e de outros codificadores fractais convencionais, onde a imagem é codificada inteira por predição fractal.

A idéia por trás da maioria dos codificadores híbridos fractais-wavelet é aplicar a DWT (*Discrete Wavelet Transform* – Transformada Discreta Wavelet) à imagem e em seguida usar métodos fractais no domínio *wavelet* [5]. Sabe-se que a concentração de energia *wavelet* está localizada primariamente no canto superior esquerdo dos valores dos filtros passa-baixas, o que torna a subbanda de aproximação extremamente favorável à aplicação de técnicas fractais. Tal característica foi explorada em [46], no qual é proposto um codificador fractal acelerado que aplica a classificação de domínios de Fisher à subbanda passa-baixas da imagem transformada via wavelet e também um codificador SPIHT (*Set Partitioning in Hierarchical Trees*) modificado nos coeficientes remanescentes.

O número bastante restrito dessas publicações sobre Codificadores Híbridos Fractal-Wavelet e o fato de seus autores não explicitarem todos os parâmetros utilizados (dada a abrangência das duas técnicas) torna difícil a comparação via implementação desses codificadores, uma vez que existe quase uma impossibilidade de reproduzir seus resultados. Em contrapartida, os codificadores híbridos se tornam um vasto campo para a pesquisa.

V. Desempenho Comparativo entre Algoritmos

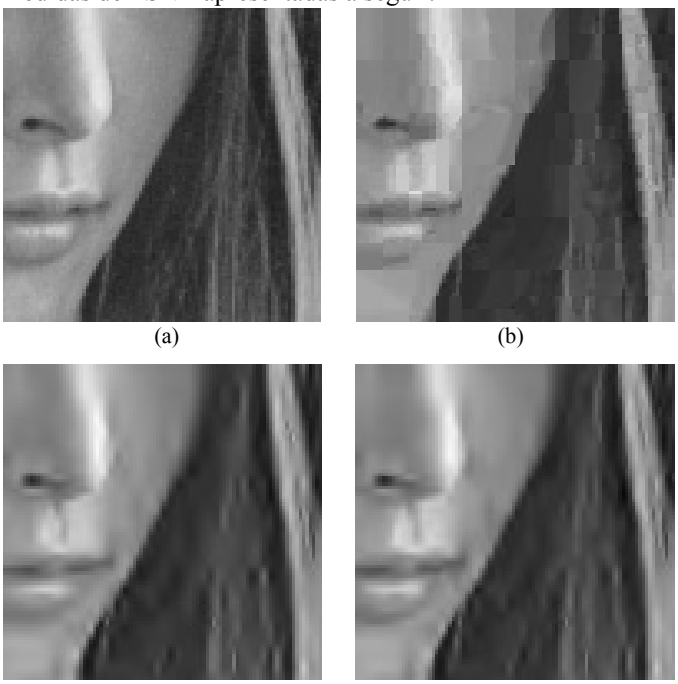
A título de ilustração, a seguir é realizada uma breve comparação entre algoritmos fractais puros e outras técnicas, incluindo técnicas híbridas (no caso, exemplificando técnicas mistas fractais e wavelets). Os codificadores fractais puros, wavelet puro, híbrido fractal-wavelet e JPEG2000 foram comparados em termos de qualidade visual, PSNR e tempo de codificação. O primeiro, o codificador fractal puro acelerado de Fisher [3], será referenciado como "QPIFS", enquanto a técnica wavelet pura será referenciada como "SPIHT". O codificador híbrido utilizado como referência ilustrativa foi extraído de [46], aonde se encontram maiores detalhes acerca das simulações.

Uma melhora significativa na qualidade subjetiva é freqüentemente observada em codificadores híbridos fractais-wavelets, evitando-se artefatos de "blurring" e blocagem. Observando a Fig. 6 (ampliação de Lena 512x512), pode-se notar que nas imagens reconstruídas pelo QPIFS artefatos de blocagem são altamente visíveis devido ao particionamento fractal por blocos.

Por outro lado, o esquema SPIHT causa evidente "blurring" em regiões como o cabelo e a boca de Lena, como conseqüência do *threshold* de quantização. O codificador híbrido usado no experimento apresenta melhoria significativa na qualidade visual da imagem reconstruída, combinando as vantagens de ambas as técnicas, evitando, desse modo, os

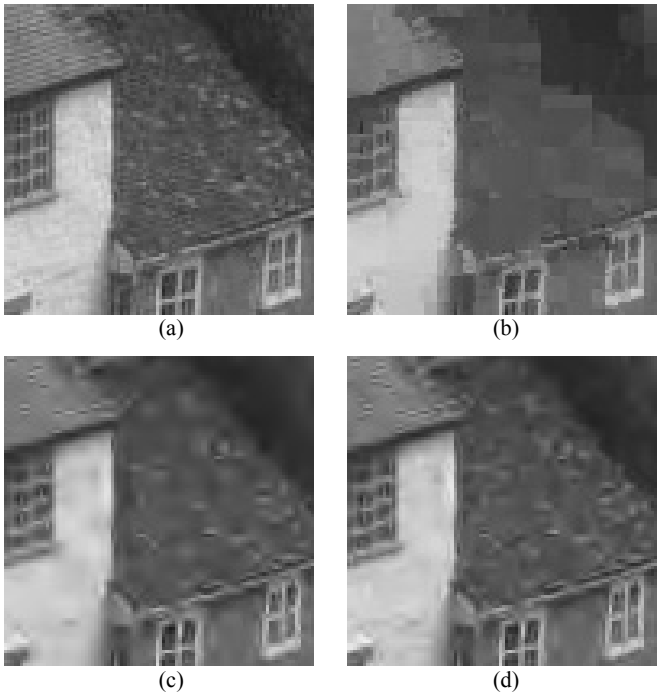
artefatos de compressão que cada técnica apresenta individualmente.

As mesmas conclusões podem ser observadas na Fig. 7, ampliação da imagem Goldhill 512x512. Resultados similares foram obtidos com o conjunto de imagens padrão em tons de cinza disponíveis no Waterloo Bragzone website[*1] a várias taxas de compressão, resultados estes corroborados nas medidas de PSNR apresentadas a seguir.



(a) (b) (c) (d)

Fig. 6 – Ampliações de Lena codificada a 0,32 bpp. (a)Original, (b)QPIFS, (c) SPIHT, (d) Híbrido



(a) (b) (c) (d)

Fig. 7 – Ampliações de Goldhill codificada a 0,32 bpp. (a)Original, (b)QPIFS, (c) SPIHT, (d) Híbrido

([*1]) http://links.uwaterloo.ca/bragzone

Com relação ao tempo de codificação, o codificador híbrido em [46] apresenta desempenho entre os esquemas SPIHT e QPIFS em todos os casos. O citado codificador híbrido apresentou uma média de 1,7 vezes o tempo de codificação do SPIHT, e o QPIFS apresentou uma média de 17 vezes o tempo de codificação do método híbrido. Dessa forma, o codificador híbrido ilustrado apresentou uma média de 94% de redução no tempo de codificação fractal puro, tempo este que pode ser consideravelmente melhorado caso sejam combinadas técnicas aceleradoras mais recentes.

O codificador JPEG2000 utilizado[*2] nas simulações, por se tratar de um software fechado, sem acesso ao código-fonte, impede comparações reais quanto ao tempo de processamento com os demais métodos. Contudo, comparações quanto às medidas de PSNR encontram-se na Fig. 8.

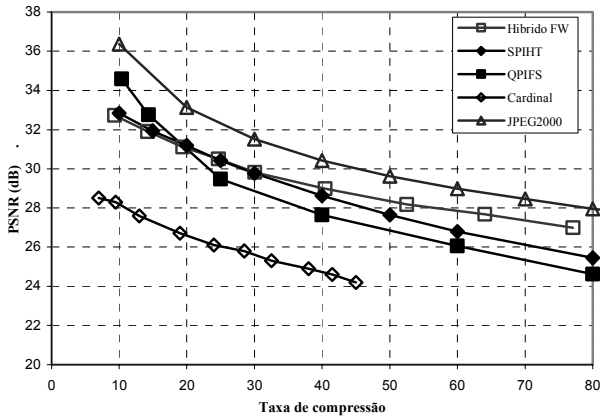


Fig. 8 – Curvas de taxa-distorção

Naturalmente o desempenho do JPEG2000 em PSNR e tempo de codificação é superior a todos os demais, por ser, mais do que um padrão já consolidado e exaustivamente testado, o principal padrão de codificação atual para imagens estáticas, fazendo uso de técnicas de compressão baseadas em tecnologia *wavelet* discreta em estado da arte e em codificação aritmética. Nenhum dos demais algoritmos utilizados na comparação contaram com codificação aritmética ou entrópica.

VI. CONCLUSÕES

É fundamental perceber que a combinação de técnicas fractais com outras técnicas, como as transformadas wavelets no exemplo apresentado, promovem um salto em desempenho na codificação fractal, tanto com relação à qualidade visual, como em medidas objetivas de PSNR e tempo de processamento, aproximando seu desempenho de padrões mais avançados e complexos como o JPEG2000.

Métodos fractais, devido ao tempo exaustivo de codificação são tradicionalmente mais convenientes para aplicações de arquivamento, tais como enciclopédias digitais, onde uma imagem é codificada uma vez e decodificada várias vezes. Este artigo procura destacar que a combinação com novas técnicas, tais como wavelets, novos codificadores entrópicos,

([*2]) http://www.luratech.com

técnicas de lifting otimizadas, hardware dedicado otimizado ou mesmo novos algoritmos de procura rápida podem permitir a extensão de técnicas fractais para tempo real.

VII. AGRADECIMENTOS

Este trabalho foi financiado pela FAPESP – Fundação de Amparo à Pesquisa do Estado de São Paulo.

REFERÊNCIAS

[1] A.L. M. C.; Y. Iano – "Procedimentos Para Método Híbrido de Compressão de Imagens Digitais Utilizando Transformadas Wavelet e Codificação Fractal" – Dissertação de Doutorado, FEEC/UNICAMP, SP, Brasil, Maio, 2005.

[2] Special Issue on the H.264/AVC Video Coding Standard –*IEEE Transactions on CSVT*, vol.13, no.7, , Jul, 2003.

[3] Y. Fisher – "Fractal Image Compression, Theory and Application" – Ed. Springer-Verlag, New York Inc, USA, 1995.

[4] N. Lu – "Fractal Imaging" – Academic Press, San Diego, USA, 1997;

[5] S. T. Wealsted – "Fractal and Wavelet Image Compression Techniques" – Ed. SPIE Optical Engineering Press, Washington, USA, 1999.

[6] K. R. Rao; P. C. Yip – "The Transform and Data Compression Handbook" – Ed. CRC Press, LLC, USA, 2001.

[7] B. Mandelbrot – "The Fractal Geometry of Nature" – Ed. W. H. Freeman and Company, New York, USA, 1983.

[8] M. F. Barnsley; A. Sloan – "A better Way to Compress Images" – Byte, pp. 215-223, Jan, 1988.

[9] M. F. Barnsley; A. Sloan – "Method and Apparatus for Image Compression by Iterated Funcion System" – US Patent # 4.941.193, 1990.

[10] M. F. Barnsley; A. Sloan – "Method and Apparatus for Processing Digital Data" – US Patent # 5.065.447, 1991.

[11] A. Jacquin – "Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations" – IEEE Transactions on Image Processing, vol. 01, no. 01, pp. 18-30, 1992.

[12] M. Peruggia– "Discrete Iterated Function Systems" – Ed. A K Peters, Massachusetts, USA, 1993.

[13] J. E. Hutchinson – "Fractal and Self-Similarity" – Indiana University Mathematics Journal, Vol. 35, No. 5, 1981.

[14] M. F. Barnsley; A. Jacquin – "Applications of Recurrent Iterated Function Systems to Images" – SPIE Visual Communications and Image Processing , pp. 122-131, 1998.

[15] E. L. Lima – "Espaços métricos" - Ed. Projeto Euclides – Rio de Janeiro, Brasil, 1993.

[16] M. F. Barnsley– "Fractals Everywhere 2nd edition" – Ed. Morgan Kaufmann, Academic Press, San Diego, USA, 1993.

[17] A. Jacquin – "A Fractal Theory of Iterated Markov Operators with Applications to Digital Image Compression" – PhD. Thesis, Georgia Institute of Technology, USA, 1989.

[18] A. Bogdan; H. Meadows – "Kohonen Neural Network for Image Coding Based on Iteration Transformation Theory" – *Proceedings of SPIE*, vol. 1766, pp. 425-436, 1992.

[19] D. McGregor; R. J. Fryer; W. P. Cookshott; P. Murray – "Fast Fractal Transform Method for Data Compression" – *University of Strachclyde Research Report*, 94/156 [IKBS-17-94], 1994.

[20] D. Saupe – "Accelerating Fractal Image Compression by Multi-Dimensional Nearest Neighbor Search" –*Proceedings DCC'95 Data Compression Conference*, pp. 222-231, Mar 1995.

[21] J. Cardinal – "Fast Fractal Compression of Greyscale Images" – *IEEE Transactions on Image Processing*, vol. 10, no. 01, pp. 159-164, Jan., 2001.

[22] J. Kominek – "Algorithm for Fast Fractal Image Compression" – *Proceedings of SPIE Symp. Electronic Imaging: Science Technology*, vol. 2419, 1995.

[23] B. E. Wohlberg; G. de Jager – "Fast Image Domain Fractal Compression by DCT Domain Block Matching" – *Electron. Lett.*, vol. 31, pp. 869-870,1995.

[24] R. F. Sproull – "Refinements to Nearest Neighbor Searching in K-Dimensional Trees" – *Algorithmica*, vol. 6, pp. 579-589, 1991.

[25] I. Katsavounidis; C.-C. J. Kuo; Z. Zhang – "Fast Tree-Structured Nearest Neighbor Encoding for Vector Quantization" – *IEEE Transactions on Image Processing*, vol. 5, pp. 398-404, Fev, 1996.

[26] C. S. Tong; M. Wong – "Adaptative Approximate Nearest Neighbor Search for Fractal Image Compression" – IEEE Transactions on Image Processing, vol. 11, No.06, pp. 605-615, Jun, 2002.

[27] C. S. Tong; M. Pi – "Fast Fractal Image Encoding Based on Adaptative Search" – IEEE Transactions on Image Processing, vol. 10, No.09, pp. 1269-1277, Set, 2001.

[28] C. -C. Wang; C. –H. Hsieh – "An Efficient Fractal Image Coding Method Using Interblock Correlation Search" – IEEE Transactions on CSVT, vol. 11, No.02, pp. 257-261, Fev, 2001.

[29] B. B. Eqbal – "Enhancing the Speed of Fractal Image Compression" – *Opt. Eng*, vol. 34, No.06, pp. 1705-1710, Jun, 1995.

[30] J. Li; G. Chen; Z. Chi – "A Fuzzy Image Metric with Application on Fractal Coding" – *IEEE Transactions on Image Processing*, vol. 11, no. 6, pp. 636-643, Jun, 2002.

[31] T. –K. Truong; J. -H, Jeng; I. S. Reed; P. C. Lee; A. Q. Li – "A Fast Encoding Algorithm for Fractal Image Compression Using DCT Inner Product" - *IEEE Transactions on Image Processing*, vol. 9, No.4, pp. 529-534, Abr, 2000.

[32] S. Lee; S. Ra – "An Analysis of Isometry Transforms in Frequency Domain for the Fast Fractal Coding" – *IEEE Signal Proc. Lett*, vol. 6, pp. 100-102, Maio, 1999.

[33] D. Saupe; H. Harteinstein – "Lossless Acceleration of Fractal Image Compression by Fast Convolution" – *Proc. IEEE Int. Conf. Image Processing*, vol. 1, pp. 185-188, Sep, 1996.

[34] Y. –G. Wu; M.-Z. Huang; Y.-L.Wen – "Fractal Image Compression with Variance and Mean" – *IEEE ICME*, vol. 1, pp. 353-356, 2003.

[35] C.-M. Lai; K.-M. Lam; W. –C. Siu– "A Fast Fractal Image Coding Based on Kick-Out and Zero Contrast Conditions" - *IEEE Transactions on Image Processing*, vol. 12, No.11, pp. 1398-1403, Nov, 2003.

[36] C. He; S. X. Yang; X. Huang– "Variance-Based Accelerating Scheme for Fractal Image Encoding" – *Eletronics Letters*, vol. 40, No.2, Jan, 2004.

[37] N. Rowshanbin; S. Samavi; S. Shirani – "Acceleration of Fractal Image Compression Using Characteristic Vector Classification" – *IEEE CCECE/CCGEI*, Ottawa, pp. 2057-2060, May, 2006.

[38] C. Caso; C. -C. J. Kuo – "New Results for Fractal/Wavelet Image Compression" – Proceedings of SPIE Visual Communications and Image Processing, vol. 2727, pp. 536-547, Mar, 1996.

[39] R. Rinaldo; G. Calvagno – "Image Coding by Block Prediction of Multiresolution Subimages" – IEEE Transactions on Image Processing, vol. 4, no. 07, pp. 909-920, Jul, 1995.

[40] G. M. Davis – "A Wavelet Based Analysis of Fractal Image Compression" – IEEE Transactions on Image Processing, vol. 7, no. 02, pp. 141-154, Fev, 1998.

[41] S. Asgari; T. Q. Nguyen; W. A. Sethares – "Wavelet-Based Fractal Transforms for Image Coding With no Search" – Proc IEEE International Conf. On Image Processing, ICIP97, 1997.

[42] D. Herbert; E. Soundarajan – "Fast Fractal Image compression With Triangulation Wavelets " – Proc. SPIE Conf. On Wavelets Applications in Signal and Image Processing VI, San Diego, USA, 1998.

[43] G. M. Davis – "Adaptative Self-Quantization of Wavelet Subtrees: A Wavelet-Based Theory of Fractal Image Compression " – Proc. SPIE Conf. On Wavelets Applications in Signal and Image Processing III, San Diego, USA, 1995.

[44] G. M. Davis – "Implicit Image Models for Fractal Image Compression " – Proc. SPIE Conf. On Wavelets Applications in Signal and Image Processing IV, Denver, USA, 1996.

[45] J. Li; C. –C. J. Kuo – "Image Compression With a Hybrid Wavelet-Fractal Coder" – IEEE Transactions on Image Processing, vol. 8, no. 06, pp. 868-874, Jun, 1999.

[46] Y. Iano, F. S. Silva, A. L.M. Cruz – "A Fast and Efficient Hybrid Fractal-Wavelet Image Coder" – IEEE Transactions on Image Processing, vol. 15, no. 1, pp. 98-105, Jan, 2006.

Ana Lúcia M. Cruz Nasceu em São Paulo, 1974. Concluiu o curso de Graduação em Engenharia Elétrica em 1998 e obteve o título de Mestrado em 2001 e de Doutorado em 2005, pela UNICAMP. Atualmente está no programa de Pós Doutoramento do Departamento de Comunicações da Faculdade de Engenharia Elétrica e Computação da Unicamp. Seus interesses incluem Processamento, compressão e codificação de Imagem e vídeo digitais, Televisão Digital e Comunicações por satélite.

Fernando S. da Silva Nasceu em São Paulo, 1973. Concluiu o curso de Graduação em Engenharia Elétrica em 1998 e obteve o título de Mestrado em 2001 e de Doutorado em 2005, pela UNICAMP. Atualmente está entrando no programa de pesquisador colaborador voluntário no Departamento de Comunicações da Faculdade de Engenharia Elétrica e Computação da Unicamp e leciona no Centro Universitário Salesiano de São Paulo (Unisal). Seus interesses incluem processamento, compressão e codificação de imagem e vídeo digitais, televisão digital e comunicações por satélite.

Yuzo Iano Nasceu em Lins, 1950. Concluiu o curso de Graduação em Engenharia Elétrica em 1972 e obteve o título de Mestrado em 1974 e de Doutorado em 1986, pela UNICAMP. Atualmente é Professor Adjunto no Departamento de Comunicações da Faculdade de Engenharia Elétrica e Computação da Unicamp e é responsável pelo Laboratório de Comunicações Visuais. Ele já desenvolveu projeto de processamento digital de sinais (áudio e imagem) em conjunto com o CPqD. Seus interesses incluem codificação de áudio e video, vídeo digital, transmissão digital de sinais, televisão digital e comunicações por satélite.

Roger F. L. Chavez concluiu o curso de Graduação em Engenharia de Sistemas na Universidade de San Agustín, Arequipa, Peru, em 2002. Recebeu seu título de mestre na Universidade Estadual de Campinas – UNICAMP – em 2007, especializado em reconhecimento de padrões. Atualmente, está no programa de doutorado no Laboratório de Comunicações Visuais do Depto. de Comunicações da Faculdade de Engenharia Elétrica e de Computação da UNICAMP. Seus interesses atuais concentram-se no desenvolvimento de um método eficiente para sincronismo de TV digital e códigos corretores de erro.