Codificação Fractal Automática de Imagens

Com a proliferação do uso de imagens digitais, devido ao crescimento dos bancos de imagens e sistemas multimídia, e com o uso cada vez mais intenso de redes, as necessidades de armazenamento e transmissão eficientes de imagens tornaram-se maiores. Várias novas formas de transmitir e armazenar imagens eficientemente têm surgido. A codificação fractal pode ser considerada uma destas novas formas. Este trabalho tem o objetivo de esclarecer as metodologias envolvidas neste tipo de codificação, aplicáveis a simples conjuntos de pontos ou às mais complexas imagens. Diversas metodologias sugeridas em outros trabalhos são comparadas, em imagens em tons de cinza. As técnicas Busca Exaustiva Leve, Sub Busca Local, Busca Local, Busca Unitária, Busca em Área Restrita, Dimensão Fractal Local foram implementadas em um mesmo programa base de compressão fractal por Busca Exaustiva. As técnicas Dupla Classificação Canônica, Classificação Canônica através das Intensidades, Dupla Classificação Canônica com Quadtree de 3 níveis foram analisadas a partir de programa anteriormente implementado. A comparação entre os método é feita considerando a qualidade visual da imagem, o tempo de codificação, o erro *rms*, a relação sinal ruído *rms* e sinal ruído de pico.

1. Introdução

A compressão de imagem visa minimizar do número de bits necessários à representação de uma imagem, o que é importante na sua transmissão e armazenamento. O tempo de transmissão e o espaço para armazenar um arquivo são proporcionais ao seu tamanho. Considere, por exemplo, uma imagem qualquer, em tons de cinza, representada por 1024x1024 pixels. Cada um destes pixels pode assumir um valor entre 0 e 255 (256 tons de cinza). Se nenhum tipo de compressão for utilizado, serão necessários 1024x1024x8 bits de memória, 8.388.608 bits, somente para armazenar esta imagem. A compressão procura reduzir o número de bits necessários para representar uma imagem, podendo esta nova representação gerar uma réplica exata (**compressão sem perdas**) ou uma cópia aproximada (**compressão com perdas)** da imagem original. A compressão fractal é uma compressão com perdas, baseada em conceitos psico-fisiológicos da visão (e interpretação) humana. Se a distorção resultante, que pode ou não estar visualmente aparente, for tolerável, a redução no tamanho do arquivo compensa o tempo gasto nas etapas de compressão e descompressão[17].

1.1 Organização deste capítulo

Na próxima seção revemos os conceitos fundamentais ao entendimento da compressão fractal de imagens. Após a apresentação dos conceitos, o processo de geração de imagens fractais através de Sistemas de Funções de Iteração é descrito e demonstrado. A compressão fractal clássica é introduzida, juntamente com o teorema da Colagem. Na seção 3, a compressão fractal é automatizada para tornar o processo de compressão

implementavél de modo eficiente sem iteração com o usuário. São utilizados Sistemas de Funções de Iteração Particionados, que permitem que regiões da imagem sejam mapeadas em outras regiões da imagem. Algumas técnicas de codificação fractal automática existentes são descritas. Os resultados do algoritmo base aplicado a uma imagem de teste é apresentado. Também são apresentados diversos métodos conhecidos, resultados de modificações que podem ser efetuadas sobre o algoritmo base. Essas modificações foram implementadas e os seus resultados são apresentadas na seção 4. Finalmente são feitos comentários sobre as vantagens, limitações, e conclusões relacionadas a desenvolvimentos futuros.

2. Conceitos Fundamentais à Compressão Fractal

Para que a compressão fractal de uma imagem seja compreendida, são necessários alguns conceitos básicos. Nesta seção apresentaremos definições e teoremas, extraídos da referência[2], essenciais ao entendimento de como "ocorre" a compressão fractal de imagens (essa teoria é melhor detalhada e exemplificada no texto que pode ser obtido em http://www.ic.uff.br/~aconci/curso/cap2.ps).

A geometria fractal preocupa-se com a estrutura de subconjuntos de espaços geométricos muito simples. Denota-se este espaço por **X** e seus elementos por *x*.

Um conjunto fechado contém todos os pontos de convergência de suas seqüências de Cauchy.

Seja (**X**, *d*) um espaço métrico completo. Então H(**X**) denota o espaço onde os pontos são subconjuntos compactos de **X**, diferentes do conjunto vazio.

A distância entre dois conjuntos não comuta (veja figura 2.2). A notação $x \vee y$ significa o máximo de dois números reais x e y.
Seja (**X**, *d*) um espaço métrico completo. Então a **distância de Hausdorff** entre os pontos A e B em $H(\mathbf{X})$ é definida por $h(A,B) = d(A,B) \vee d(B,A)$
Esta distância obedece aos axiomas da definição 2.2, podendo ser considerada uma **métrica** para $H(\mathbf{X})$.

Alguns autores referem-se ao espaço métrico ($H(\mathbf{X})$, $h(d)$) como "o espaço dos fractais" [2]. De um modo amplo, qualquer subconjunto de ($H(\mathbf{X})$, $h(d)$) é um fractal [2].

Seja $S \subset \mathbf{X}$ e seja $\Gamma \geq 0$. Então $S_{+\Gamma}=\{y \in \mathbf{X}: d(x,y) \leq \Gamma$ para algum $x \in S\}$. $S_{+\Gamma}$ é algumas vezes chamado, por exemplo, na **teoria de morfologia de conjuntos** [17], de **dilatação** de S por uma bola de raio Γ.

Lema 1: Seja A e B dois conjuntos pertencentes a $H(\mathbf{X})$ onde (**X**, *d*) é um espaço métrico. Seja $\varepsilon > 0$. Então $h(A,B) \leq \varepsilon \Leftrightarrow A \subset B_{+\varepsilon}$ e $B \subset A_{+\varepsilon}$.

Lema2: Lema de Extensão. Seja (**X**, *d*) um espaço métrico. Seja $\{A_n : n = 1, 2, ..., \infty\}$ uma seqüência de Cauchy de pontos em ($H(\mathbf{X})$, $h(d)$). Seja $\{n_j\}$, $j = 1, 2, ..., \infty$, uma seqüência infinita de inteiros: $0 < n_1 < n_2 < n_3 < ...$ .Suponha que exista uma seqüência de Cauchy $\{x_{nj} \in A_{nj} : j = 1, 2, ...\}$ em (**X**, *d*). Então existe uma seqüência de Cauchy $\{x^s_n \in A_n : n = 1, 2,...\}$ tal que $x^s_{nj} = x_{nj}$, para todo j=1,2,....

Teorema : Seja (**X**, *d*) um espaço métrico completo. Então (*H*(**X**),*h*(*d*)) é um **espaço métrico completo**. Além disso, se $\{A_n \in H(\mathbf{X})\}$, $n=1,2,...,\infty$, é uma seqüência de Cauchy, então $A=\lim_{n\to\infty} A_n \in H(\mathbf{X})$ pode ser caracterizado da seguinte forma:

$A=\{x \in \mathbf{X}$: existe uma seqüência de Cauchy, $x_n \in A_n$, que converge para $x\}$.

Teorema : Se **R** representa o conjunto dos reais, então a função $f:\mathbf{R}\to H(\mathbf{X})$ dado por $f(x)=\{x\}$ é contínua.

O teorema acima mostra que existe um caminho em *H* de um intervalo até um de seus limites extremos.

Teorema : Se *A* é um subconjunto compacto dos reais, **R**, então a função $f_A : [0 , b] \to H(\mathbf{X})$ dada por $f_A(a) = \cup [x, x + a]$ tal que $x \in A$ é contínua.

Teorema : Se *A* é um subconjunto compacto de **R**, então o conjunto $\cup [x, x + b]$ tal que $x \in A$ é um intervalo grande o suficiente para *b*.

Teorema : Se *A* e *B* são subconjuntos compactos dos reais, **R**, então existe um caminho em *H*(**R**) conectando *A* e *B*.

Teorema : Seja (**X**, *d*) um espaço métrico. Seja $\{x_n\}$ uma seqüência de Cauchy convergente para $x \in \mathbf{X}$ (ou, de forma equivalente, seja $\{x_n\}$ a seqüência e seja *x* um ponto, tal que $\lim_{n\to\infty} d(x, x_n) = 0$). Seja $f : \mathbf{X} \to \mathbf{X}$ contínua. Então $\lim_{n\to\infty} f(x_n) = f(x)$.

Teorema 2.10: Sejam ($\mathbf{X}_1$, d_1) e ($\mathbf{X}_2$, d_2) espaços métricos. Seja $f : \mathbf{X}_1 \to \mathbf{X}_2$ uma função contínua e $E \subset \mathbf{X}_1$ compacto. Então $f: E\to\mathbf{X}_2$ é uniformemente contínua: ou seja, dado $\varepsilon>0$ existe um número $\delta>0$ tal que; $d_2(f(x),f(y)) < \varepsilon$ sempre que $d_1(x,y) < \delta$ para todo $x, y \in E$.

Espaços $\mathbf{X}_1$ e $\mathbf{X}_2$ podem ser usados para criar um novo espaço denotado por $\mathbf{X}_1 x \mathbf{X}_2$, chamado de produto Cartesiano de $\mathbf{X}_1$ e $\mathbf{X}_2$. Um ponto em $\mathbf{X}_1 x \mathbf{X}_2$ é representado pelo par ordenado (x_1,x_2), onde $x_1 \in \mathbf{X}_1$ e $x_2 \in \mathbf{X}_2$. Por exemplo, $\mathbf{R}^2$ é o produto Cartesiano de **R** e **R**.

Teorema : Sejam ($\boldsymbol{X}_i$, d_i) espaços métricos, para $i = 1, 2, 3$. Seja $f : \mathbf{X}_1 \times \mathbf{X}_2 \to \mathbf{X}_3$ uma função com a seguinte propriedade: para cada $\varepsilon > 0$ existe $\delta > 0$ tal que

$$d_1(x_1,y_1) < \delta \Rightarrow d_3(f(x_1, x_2), f(y_1, y_2)) < \varepsilon, \forall\, x_1, y_1 \in \mathbf{X}_1 \;\forall\, x_2, y_2 \in \mathbf{X}_2$$

e $$d_2(x_2,y_2) < \delta \Rightarrow d_3(f(y_1, x_2), f(y_1, y_2)) < \varepsilon \;\forall\, x_1, y_1 \in \mathbf{X}_1 \;\forall\, x_2, y_2 \in \mathbf{X}_2$$

Então *f* é contínua no espaço métrico ($\mathbf{X}=\mathbf{X}_1 x \mathbf{X}_2$, *d*), onde a métrica para o espaço produto cartesiano é $d((x_1,x_2),(y_1, y_2)) = \max\{ d_1(x_1, y_1) , d_2(x_2, y_2) \}$.

Teorema: Sejam ($\boldsymbol{X}_i$, d_i) espaços métricos, para $i = 1, 2$ e seja o espaço métrico (**X**, *d*) definido como no teorema anterior. Se $K_1 \subset \mathbf{X}_1$ e $K_2 \subset \mathbf{X}_2$ são compactos, então $K_1 \times K_2 \subset \mathbf{X}$ é **compacto**.

Definição : Seja (**X**, *d*) um espaço métrico. Uma **transformação** em **X** é uma função $f : \mathbf{X}\to\mathbf{X}$, que fornece exatamente um ponto $f(x) \in \mathbf{X}$ para cada ponto $x \in \mathbf{X}$. Se $S \subset \mathbf{X}$, então $f(S) = \{ f(x) : x \in S \}$. *f* é **injetiva** se $x, y \in \mathbf{X}$ com $f(x) = f(y)$ implicar em $x = y$. *f* é **sobrejetiva** se $f(\mathbf{X}) = \mathbf{X}$. *f* é chamada **inversível** se for injetiva e sobrejetiva. Neste caso é possível definir uma transformação $f^{-1}: \mathbf{X}\to\mathbf{X}$, chamada de **inversa** de *f* , como: $f^{-1}(y) = x$, onde $x \in \mathbf{X}$ é um ponto único tal que $y = f(x)$.

Definição : Seja f :$\mathbf{X}\rightarrow\mathbf{X}$ uma transformação em um espaço métrico. **Iterações direta** de f são transformações definidas como $f^{o0}(x)=x, f^{o1}(x)=f(x), f^{o(n+1)}(x)=f$ o $f^{(n)}(x)=f(f^{(n)}(x))$ para n = 0, 1, 2, ... Se f é inversível, então as **iterações no sentido contrário** de f são transformações $f^{o(-m)}(x)$:$\mathbf{X}\rightarrow\mathbf{X}$ definida por $f^{o(-1)}(x)=f^{-1}(x), f^{o(-m)}(x)=(f^{om})^{-1}(x)$ para m = 1, 2, 3, ...

Definição: Uma transformação w:$\mathbf{R}^2\rightarrow\mathbf{R}^2$ com a forma $w(x_1,x_2)=(ax_1+bx_2+e,\ cx_1+dx_2+f)$, onde a, b, c, d, e, e f são números reais, é chamada de **transformação afim bidimensional**. As seguintes notações são equivalentes:

$$w(x) = w\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\cdot\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} = Ax + t \qquad \text{sendo} \quad t = \begin{pmatrix} e \\ f \end{pmatrix} \quad \text{e} \quad A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Se $(r_1\ ,\ \theta_1)$ forem as coordenadas polares do ponto (a , c) e $(r_2\ ,\ (\theta_2 + \pi/2))$ forem as coordenadas polares do ponto (b , d), então a matriz A pode ser re-escrita como

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} r_1\cos\theta_1 & -r_2\,\mathrm{sen}\,\theta_2 \\ r_1\,\mathrm{sen}\,\theta_1 & r_2\cos\theta_2 \end{pmatrix}$$

Definição : Uma transformação w:$\mathbf{R}^2\rightarrow\mathbf{R}^2$ é chamada **similitude** se ela for uma transformação afim com uma das seguintes formas: (r_1=r_2=r , θ_1=θ_2=θ ou θ_1=-θ_2=θ) *i.e.:*

$$w\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} r\cos\theta & -rsen\theta \\ rsen\theta & r\cos\theta \end{pmatrix}\cdot\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} \quad \text{ou} \quad w\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} r\cos\theta & rsen\theta \\ -rsen\theta & r\cos\theta \end{pmatrix}\cdot\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}$$

para alguma translação $(e\ ,f)\in\mathbf{R}^2$, algum número real $r\neq0$, e algum ângulo θ, $0 \le \theta < 2\pi$. θ é chamado de ângulo de rotação, enquanto r é chamado de fator de escala. A transformação linear

$$R_\theta\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \cos\theta & -sen\theta \\ sen\theta & \cos\theta \end{pmatrix}\cdot\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

é chamada **rotação**. Chama-se **reflexão** as transformações lineares:

$$R\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}\cdot\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \qquad R\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}\cdot\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \qquad R\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}\cdot\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Quando são descritas transformações em espaços, utiliza-se, freqüentemente, um sistema de **coordenadas subjacente**. Este sistema de coordenadas é um subconjunto de um espaço métrico, denotado por $\mathbf{X}_c$ Deve-se fazer distinção entre um ponto $x \in \mathbf{X}$ e sua coordenada $x \in \mathbf{X}_c$ .Pois, o espaço $\mathbf{X}_c$ pode conter pontos (coordenadas) que não correspondam a nenhum ponto no espaço $\mathbf{X}$.

Os pontos fixos de uma transformação são muito importantes, pois dizem quais partes do espaço estão "presas", ou não movidas, pela transformação. Os pontos fixos de uma transformação restringem o movimento do espaço à transformações de deformação limitada, e não violentas.

Definição: Seja f :$\mathbf{X}\rightarrow\mathbf{X}$ uma transformação em um espaço métrico. Um ponto $x_f \in \mathbf{X}$ tal que $f(x_f)=x_f$ é chamado de **ponto fixo da transformação**.

Definição: Uma transformação $f : \mathbf{X} \to \mathbf{X}$ em um espaço métrico $(\mathbf{X}, d)$ é chamada contrativa ou um **mapeamento de contração** se existir uma constante $0 \leq s < 1$ tal que $d(f(x), f(y)) \leq s \cdot d(x,y)$, $\forall x,y \in \mathbf{X}$. s é chamado de fator de contração de f.

Teorema : (Teorema do Mapeamento de Contração) Se $f: \mathbf{X} \to \mathbf{X}$ é um mapeamento de contração em um espaço métrico completo $(\mathbf{X}, d)$. Então f possui exatamente um ponto fixo $x_f \in \mathbf{X}$ e, além disso, para cada ponto $x \in \mathbf{X}$, a seqüência $\{f^{on}(x): n = 0, 1, 2, ...\}$ converge para x_f. Ou seja: $\lim_{n \to \infty} f^{on}(x) = x_f$, para cada $x \in \mathbf{X}$.

Lema : Se $w: \mathbf{X} \to \mathbf{X}$ é um mapeamento de contração no espaço métrico $(\mathbf{X}, d)$. Então w é contínuo.

Lema : Se $w: \mathbf{X} \to \mathbf{X}$ é um mapeamento contínuo no espaço métrico $(\mathbf{X}, d)$. Então w mapeia $H(\mathbf{X})$ nele mesmo.

Lema : Se $w: \mathbf{X} \to \mathbf{X}$ é um mapeamento de contração no espaço métrico $(\mathbf{X}, d)$ com fator de contração s. Então $w: H(\mathbf{X}) \to H(\mathbf{X})$ definido por $w(B) = \{w(x): x \in B\} \forall B \in H(\mathbf{X})$ é um mapeamento com contração em $(H(\mathbf{X}), h(d))$ com fator de contração s.

Lema : Para todo B, C, D, e E, em $H(\mathbf{X})$ tem-se que $h(B \cup C, D \cup E) \leq h(B, D) \vee h(C, E)$, onde h é a métrica de Hausdorff.

O lema a seguir fornece um método importante para combinar mapeamentos de contração em $(H(\mathbf{X}), h(d))$ para produzir novos mapeamentos com contração em $(H(\mathbf{X}), h(d))$. Este método é diferente da composição de mapeamentos.

Lema : Seja $(\mathbf{X}, d)$ um espaço métrico. Sejam $\{w_n : n = 1, 2, ..., N\}$ mapeamentos de contração em $(H(\mathbf{X}), h(d))$. Se o fator de contração para w_n for denotado por s_n para cada n, definindo $W : H(\mathbf{X}) \to H(\mathbf{X})$ como

$$W(B) = w_1(B) \cup w_2(B) \cup ... \cup w_n(B)$$
$$= \cup w_i(B) , i = 1, 2,..., n ; \text{ para cada } B \in H(\mathbf{X}).$$

Então W é um mapeamento de contração com fator de contração $s = \max\{s_n: n = 1, 2, ..., N\}$.

Definição: Um sistema de **funções de iteração hiperbólico** consiste de um espaço métrico $(\mathbf{X}, d)$ junto com um conjunto finito de mapeamentos de contração $w_n: \mathbf{X} \to \mathbf{X}$, com os respectivos fatores de contração s_n , para $n = 1, 2, ..., N$. A abreviação "**SFI**" é utilizada para sistema. $\{ \mathbf{X} ; w_n : n = 1 , 2 , ... , N \}$ é a notação usada para um SFI , sendo $s = \max \{ s_n : n = 1, 2, ..., N \}$ seu fator de contração.

Teorema : Seja $\{ \mathbf{X} ; w_n : n = 1, 2, ..., N \}$ um sistema de funções de iteração hiperbólico com fator de contração s. Então a transformação $W : H(\mathbf{X}) \to H(\mathbf{X})$ definida por: $W(B) = \cup w_i(B) , i = 1, 2,..., n$ para todo $B \in H(\mathbf{X})$, é um **mapeamento de contração** no espaço métrico completo $(H(\mathbf{X}) , h(d))$ com **fator de contração** s. Sendo $h (W(B), W(C)) \leq s \cdot h (B, C)$ para todo $B, C \in H(\mathbf{X})$. Seu **ponto fixo** único, $A \in H(\mathbf{X})$, obedece $A = W(A) = \cup w_i(A) , i = 1, 2,..., n$ e é dado por $A = \lim_{n \to \infty} W^{on}(B)$ para qualquer $B \in H(\mathbf{X})$.

Definição : O ponto fixo $A \in H(\mathbf{X})$ do teorema acima é chamado de **atrator do SFI.**

O teorema seguinte tem extrema importância no projeto de SFI que possuem atratores que se aproximam de conjuntos dados.

Teorema: (**Teorema da Colagem**). Seja $(\mathbf{X}, d)$ um espaço métrico completo. Sejam $L \in H(\mathbf{X})$ e $\varepsilon \geq 0$ dados. Escolha um SFI $\{\mathbf{X}; w_1, w_2, ..., w_n\}$ com fator de contração $0 \leq s < 1$, tal

que $h(L;\cup w_i(L),\ i=1,2,...,n)\leq\varepsilon$, onde $h(d)$ é a métrica de Hausdorff. Então $h(L, A)\leq\varepsilon/(1-s)$, onde A é o atrator do SFI. Equivalentemente, $h(L, A) \leq (1-s)^{-1}.h(L,\cup w_i(L)\ ,\ i=1,2,...,n)$, para todo $L\in H(\mathbf{X})$.

Este teorema mostra que para se encontrar o SFI cujo atrator é "uma aproximação" ou "se parece" com um dado conjunto, tem-se que tentar encontrar um conjunto de transformações (mapeamentos de contração no espaço onde se encontra o conjunto dado) tal que a **união das imagens do conjunto** dado sob as transformações esteja próxima ao conjunto dado. A proximidade é medida usando a **métrica de Hausdorff.**

Lema: Seja ($\mathbf{X}$, d) um espaço métrico completo. Se $f:\mathbf{X}\rightarrow\mathbf{X}$ for um mapeamento de contração com fator de contração $0\leq s<1$, e se o **ponto fixo** de f for $x_f\in\mathbf{X}$. Então $d(x, x_f) \leq (1-s)^{-1}.d(x, f(x))$, para todo $x\in\mathbf{X}$.

2.1 Construção de imagens através de um SFI

Para gerarmos um conjunto através de um sistema de funções de iterações conhecido podemos escolher entre duas estratégias, a **determinística** e a **aleatória**. Os algoritmos para as duas estratégias são bastante simples e podem ser descritos como: Se $\{\mathbf{X}; w_1,w_2,..., w_N\}$ for um sistema de funções de iteração hiperbólico, a partir do qual se quer gerar um atrator, escolha um conjunto compacto $A_0 \subset \mathbf{R}^2$ qualquer. Então calcule sucessivamente $A_n = W^{on}(A)$ de acordo com: $A_{n+1} = \cup\ w_j\ (A_n)$, j = 1, 2, ..., n ; para n = 1, 2, ... Assim, construa a seqüência $\{A_n : n = 1, 2, 3, ...\} \subset H(\mathbf{X})$. Então, pelo teorema 2.15, a seqüência $\{A_n\}$ converge para o atrator do SFI na métrica de Hausdorff [2].

Algoritmo Determinístico para a Geração de Imagens em Preto e Branco
Início
Defina um conjunto (semente) inicial A_0;
Defina os coeficientes das transformações afins: a_i, b_i, c_i, d_i, e_i, f_i; i = 1, 2,..., N,
onde N = número de transformações afins;
Plote o conjunto A_0;
Para k = 0 até k = número de iterações Faça
Aplique todas as transformações ao conjunto A_k ;
Limpe a tela;
Plote o conjunto A_{k+1} ;
Fim do Para
Fim

O Teorema do Mapeamento de Contração (Teorema 2.14) garante que partindo de qualquer conjunto inicial (semente) será encontrado sempre o mesmo atrator para um determinado SFI. Como exemplo, este teorema será aplicado na construção de um conjunto muito conhecido, o Triângulo de Sierpinski.

Considerando $\mathbf{X}=[0,1]^2$, parte-se de um conjunto inicial A_0={quadrado de lados=1} e o SFI formado pelas seguintes transformações: $w_1(x,y)=(½x\ ,\ ½y)$, $w_2(x,y)=(½x+½\ ,\ ½y)$, $w_3(x,y)=(½x+¼\ ,\ ½y+½)$, usando o algoritmo determinístico aplica-se recursivamente estas transformações. A figura 2.3 mostra as primeiras iterações do processo de construção do atrator.

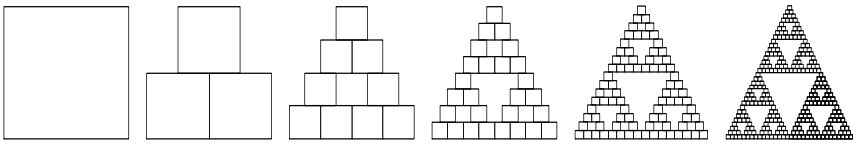


Figura 2.3 Gerando o Triângulo de Sierpinski

A compressão de uma imagem gerada por um algoritmo fractal pode ser "perfeita" (como a do exemplo da figura 3.2), chegando a taxas de compressão extremas. Pela característica de um fractal ser auto-similar em diferentes escalas, uma imagem de 256x256 pixels tem o mesmo código SFI de uma imagem de 512x512 pixels ou qualquer outra dimensão. De acordo com esta consideração imagens de maiores dimensões terão taxas de compressão mais altas que as imagens de menores dimensões. O Triângulo de Sierpinski é gerado por um SFI com apenas três transformações. Considerando que cada transformação possui 6 parâmetros, tem-se então uma imagem codificada por 18 parâmetros, ou, pensando em tamanho de arquivo de imagem, e considerando que cada parâmetro pode ser definido por um byte, tem-se 18 bytes. Se o conjunto estiver sendo mostrado como uma imagem de 512 x 512 pixels obtém-se 262.144 **bits** ou pixels, ou 32.768 bytes sendo codificados com apenas 18 bytes, ou seja, uma taxa de compressão de 1.820,44/1, ou 0,0005493 bpp (bits por pixel). Caso o conjunto fosse de uma imagem de 256 x 256 pixels (65.536 pixels ou bits, ou 8.192 bytes) a taxa de compressão seria de 455,11/1 ou 0.0021972 bpp.

Para o segundo algoritmo precisamos incluir a ideia de probabilidade às w_i do SFI. Assim se $\{\mathbf{X}; w_1, w_2, ..., w_N\}$ for um SFI hiperbólico, onde uma probabilidade $p_i>0$ foi definida para cada w_i, para i=1, 2, ..., N, sendo $\sum p_i = 1$, i = 1, 2, ..., n. Para gerar um atrator, respeitando a probabilidade de cada tranformação, escolha um ponto qualquer $x_0 \in \mathbf{X}$ e recursivamente e aplique cada uma das transformações escolhidas de acordo com suas probabilidades, $x_n \in \{w_1 (x_{n-1}), w_2 (x_{n-1}), ..., w_N (x_{n-1})\}$, para n = 1, 2, 3,...onde a probabilidade do evento $x_n=w_i (x_{n-1})$ é p_i. Construa a seqüência $\{x_n : n=1, 2, 3,...\} \subset \mathbf{X}$. Essa seqüência $\{x_n : n = 1, 2, 3, ..., \infty\}$ converge para o atrator do SFI [2].

Com o algoritmo aleatório pode-se gerar uma imagem binária, e com algumas pequenas modificações pode-se gerar imagens em tons de cinza, também conhecidas como imagens em gray-scale. Essas modificações são apresentadas logo depois.

A probabilidade associada a cada transformação determinará a maior ou menor concentração de pixels na região de atuação de cada transformação. Podemos com isso obter alguns efeitos interessantes em imagens binárias. Dependendo da concentração de pixels em uma determinada região, esta parecerá ter uma tonalidade de cinza mais forte ou mais fraca.

O Teorema da Colagem dá uma boa indicação de quais sejam as áreas de atuação de cada transformação de um SFI. Os SFIs sem probabilidades associadas as suas transformações, quando submetidos a este algoritmo aleatório, recebem um mesmo valor de probabilidade, para cada uma de suas transformações.

Algoritmo Aleatório para a Geração de Imagens em Preto e Branco
Início
Defina um ponto (semente) inicial (x_0, y_0);
Defina os coeficientes das transformações afins: a_i, b_i, c_i, d_i, e_i, f_i, e as suas respectivas probabilidades p_i ; i=1,2,...,*N*, onde *N*=número de transformações afins;
Para k = 0 até k = número de iterações Faça
Gere um número aleatório, *Na*;
Aplique a transformação associada ao número aleatório gerado;

$$x_{k+1} = a_i\, x_k + b_i\, y_k + e_i\ ; \qquad y_{k+1} = c_i\, x_k + d_i\, y_k + f_i\ ;$$

Plote o ponto $(x_{k+1}\ , y_{k+1})$;
Fim do Para
Fim

Algoritmo para a Geração de Imagem em Tons de Cinza através de um SFI.

Este algoritmo é basicamente o mesmo aleatório anterior, mas como a intenção do algoritmo é gerar uma imagem em tons de cinza, ele é mais apropriado à SFIs com probabilidades associadas, ou seja, com a adição de um parâmetro de probabilidade à cada transformação. Este novo parâmetro determinará a maior chance de uma das transformações ser mais vezes aleatóriamente escolhida que as outras. Cada vez que um pixel, a ser aceso, é definido pelo algoritmo este sofrerá uma variação na tonalidade de cinza corrente com a adição de um valor (*offset*) pré-determinado.

A mesma imagem de 256x256 da figura 2.3 será agora codificada com 21 parâmetros, 7 para cada transformação. A imagem, agora, terá 65.536 pixels, ou **bytes**, codificados por 21 bytes, ou seja, uma taxa de compressão de 3.120,76/1 ou 0.0025634 bpp (bits por pixel). Os mesmos parâmetros da imagem de 256 x 256 pixels codificarão a imagem de 512 x 512 pixels (262.144 pixels ou bytes), tem-se, então, uma taxa de compressão de 12.483,05/1 ou 0,0006408 bpp.

A imagem em tons de cinza gerada por este algoritmo sofre vários tipos de limitações, pois esta implementação, baseada na associação da intensidade do pixel com a densidade de pontos no atrator, mostra-se pouco flexível (êm a propensão de deixar vazios na imagem do atrator) O ponto principal em todos os algoritmos de geração é o conhecimento do SFI; e em algumas imagens torna-se difícil o estabelecimento de auto-similaridades entre a imagem inteira e partes dela (o que é fundamental para a determinação das w_{is}), a próxima seção trata desse ponto.

2.2 Construção de um SFI que gere uma dada imagem: o problema inverso

Uma imagem fractal pode ser vista como uma **união de cópias reduzidas** dela mesma. Cada cópia é obtida pela aplicação de um mapeamento de contração w_i de um Sistema de Funções de Interação - SFI. Restringindo-se aos SFIs compostos por transformações afins no plano Euclideano, os coeficientes da matriz **A** e do vetor coluna **t** poderão ser determinados através da simples determinação da posição onde três pontos da imagem original serão mapeados para três pontos em cada cópia reduzida.

Observando o Triângulo de Sierpinski (figura 3.2) pode-se notar que ele é formado por 3 cópias reduzidas dele mesmo, ou também, por 9 cópias reduzidas. Aplicando-se o Teorema da Colagem no Triângulo de Sierpinski, obtém-se as transformações e o código SFI dele. O código SFI é formado pelos parâmetros das transformações obtidas pelo Teorema da Colagem. Por esse teorema uma imagem semelhante (Atrator) pode ser obtida cobrindo-se a imagem que será codificada com cópias reduzidas dela mesma, e quanto melhor essa cobertura, mais semelhante será a imagem obtida. Será utilizada uma transformação para cada cópia reduzida da colagem, indicando onde e como esta cópia será posicionada. Escolhe-se, geralmente, o menor número de cópias que ofereçam uma boa colagem da imagem. Neste caso, 3 cópias são usadas para obter-se as w_i que geram a figura 2.3 e fornecerão uma colagem "perfeita", sem perdas, do Triângulo de Sierpinski.

O problema geral é determinar quão bem os conjuntos e imagens podem ser aproximados por atratores de SFI. Ou em outras palavras: dado um conjunto "alvo" S, pode ser encontrado o SFI W que possua um atrator A que aproxime este conjunto S com uma precisão desejada em uma métrica d apropriada? Ou, ainda, dado um conjunto alvo S e um número $\varepsilon > 0$, pode ser encontrado o SFI W tal que $d(S, W(S)) < \varepsilon$?
O termo $d(S, W(S))$ é conhecido como distância da colagem de uma imagem a esta imagem. Quanto menor for esta distância mais próximo $W(S)$ estará de S, e S poderá ser melhor aproximado com cópias reduzidas dele mesmo [2,3].

Sumariando a compressão de uma imagem por fractais é representada pelo seu SFI. O problema de comprimir uma dada imagem é o de **encontrar seu SFI**. O ponto chave do uso do SFI para a geração de fractais é encontrar essas transformações de contração afim. A forma de fazer **manualmente** isso para qualquer imagem a ser comprimida (mesmo que não tenha estas transformações tão obvias, como em muitas imagens fractais) é a seguinte:

1. Faça um esboço de poucas linhas da imagem a ser gerada (envolva-a por poligonais por exemplo).
2. Cubra esta imagem com cópias reduzidas da própria imagem (a mesma poligonal mas em escalas reduzidas diversas por exemplo). É importante reduzir ao mínimo as superposições de copias e os espaços vazios, para minimizar a distancia de Hausdorff entre os dois conjuntos.
3. Determine os mapeamentos que levam a imagem original em cada uma das pequenas cópias. Este procedimento é feito produzindo cada um dos mapeamentos. O conjunto das transformações afins produzirão determinado atrator que deve gerar, mais ou menos bem, a imagem dada (a aproximação pode ser determinada pelo Teorema da Colagem).

Obs. O conjunto de pontos da imagem original e do atrator, não precisam estar apenas no plano real, pode ser no plano complexo, no espaço Euclidiano ou em qualquer outro espaço de Hausdorf. E exemplo em outros espaços podem ser vistos em http://www.ic.uff.br/~aconci/compressao/fractal.html (ou fazendo *download* do arquivo: http://www.ic.uff.br/~aconci/curso/cap3.ps).

3. Automatizando a Determinação das Transformações

Determinar um mapeamento ótimo através da observação de auto-similaridades geométricas em um conjunto qualquer é um problema de difícil solução, especialmente se

deseja-se **automatizar** este processo. Para resolver o problema de automatização do processo de codificação deve-se observar que, em uma imagem, existem mais informações que simples formas geométricas. Existem variações de tons de cinza: cada pixel pode assumir um valor em um conjunto não negativo e finito.

No Sistema de Funções de Iteração, SFI, cada uma das funções do sistema mapeia uma cópia reduzida, por um fator s_i , do conjunto (imagem) semente para um ponto determinado. Apresenta-se agora um novo sistema de funções de iteração, que ao invés de partir de uma imagem semente e fazer cópias reduzidas desta, parte de regiões de uma imagem e os mapeia em regiões menores desta mesma imagem. Este sistema recebe o nome de **Sistema de Funções de Iteração Particionado** (SFIP), devido a peculiaridade de utilizar somente partes da imagem.

O SFIP possui como principal característica o estabelecimento de correlações de longa distância entre partes da imagem. Esta correlação é que define a característica fractal da imagem, estabelecendo auto-similaridades entre partes em diferentes escalas da imagem. Caso as dimensões das regiões sejam preestabelecidas e fixas, tem-se o conhecimento do número de funções que comporão o sistema, pois, como descreve-se adiante, a imagem será dividida em um número de regiões que dependerá somente do tamanho escolhido para estas regiões e as dimensões da própria imagem. Este mapeamento é chamado de não adaptativo, pois não permite uma variação dinâmica nas dimensões dos blocos.

Na compressão automática, utiliza-se SFIP que buscam a auto-similaridade entre partes maiores e partes menores da imagem. Desta forma, as imagens são vistas como uma colagem de partes auto-similares que podem ser mapeadas entre si. Como partes menores toma-se blocos quadrados de n x n pixels, chamados de blocos imagem ou molde, e como partes maiores blocos com o dobro das dimensões do menor (2n x 2n), chamados de blocos domínio. A figura 3.1 mostra estes blocos. Os blocos são tradicionalmente escolhidos com forma quadrada por ter, este tipo de particionamento, um custo computacional relativamente mais baixo.

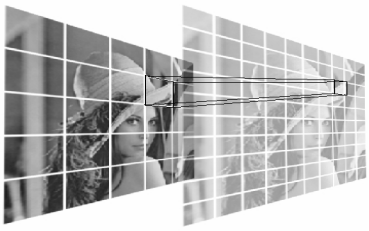


Figura 3.1 Formação de um par de blocos domínio-molde ótimo.

Nesta compressão, os sistemas de funções de iteração vistos na seção anterior são estendidos para incluir, de forma eficaz, a variação de tonalidades de uma imagem. A imagem deve ser entendida como um objeto tridimensional. O valor da tonalidade de um pixel (z) é tratado como sendo a terceira dimensão. Os blocos tornam-se, na realidade, cubos ou figuras espaciais mais complexas [11].

O Teorema do Mapeamento de Contração é também aplicado a esta terceira dimensão, garantindo-se assim a convergência das tonalidades dos pixels. A transformação afim

abaixo será acrescentada ao sistema de funções de iteração bidimensional, tornando este tridimensional:

$$w(z) = s \,.\, z + o \tag{3.1}$$

onde: s - é o coeficiente de variação do contraste. De modo que se:
s = 1 não há modificação no contraste;
s > 1 o contraste do bloco da imagem em questão é aumentado;
s < 1 o contraste do bloco da imagem em questão é diminuido;
s = 0 este bloco da imagem torna-se preto.
o - é o coeficiente de deslocamento (*offset*) da intensidade do pixel, se:
o = 0 não há deslocamento na intensidade do pixel;
o > 0 o pixel tem a sua intensidade aumentada (eixo z);
o < 0 o pixel tem a sua intensidade diminuída, sofrendo um deslocamento em z.
Cada uma das transformações afins tridimensionais toma a seguinte forma:

$$w(x) = w\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & s \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} e \\ f \\ o \end{pmatrix} = Ax + t \tag{3.2}$$

Considerando a imagem um ente tridimensional ($\in \mathbf{R}^3$), se os mapeamentos de um sistema de funções de iteração, w_i , são de contração, então quaisquer dois pontos no plano da imagem, depois de transformados, movem-se de forma a se aproximar espacialmente e em suas tonalidades.

$$d_{\text{tonal}}(w_i(u), w_i(v)) < s_{\text{i, tonal}} \,.\, d_{\text{tonal}}(u, v), \quad 0 \le s_{\text{i, tonal}} < 1 \tag{3.3}$$

$$d_{\text{geom}}(w_i(u), w_i(v)) < s_{\text{i, geom}} \,.\, d_{\text{geom}}(u, v), \quad 0 \le s_{\text{i, geom}} < 1 \tag{3.4}$$

onde: u, v - são dois pontos quaisquer;
$s_{\text{i, tonal}}$ - é o fator de contração das tonalidades (no eixo z) do mapeamento;
$s_{\text{i, geom}}$ - é o fator de contração espacial do mapeamento;
d_{tonal} - é a métrica usada para as tonalidades;
d_{geom} - é a métrica usada espacialmente.

Observando o exposto acima, tem-se que uma transformação tridimensional de um SFIP pode ser divida em duas outras, uma transformação espacial, e outra transformação nos níveis de tons de cinza. Considerando o espaço das imagens discretas, onde uma região da imagem é considerada um ponto, o teorema do mapeamento com contração garante a convergência do sistema a um "ponto fixo" (uma imagem atratora).

O código de representação SFIP de uma imagem, na implementação feita na referência [3], consiste de uma sequência de tuplas, uma por bloco:

$$w_i = (e_i, f_i, m_i, o_i, s_i) \tag{3.5}$$

onde: e_i - determina a translação do bloco domínio reduzido no eixo x;
f_i - determina a translação do bloco domínio reduzido no eixo y;
m_i - define a simetria do bloco domínio de menor distância do bloco corrente;
o_i - determina o deslocamento que o bloco domínio deverá receber no eixo z (mudança do valor da intensidade média);
s_i - determina o fator de contração espacial do mapeamento.

3.1 Considerações sobre a partição da imagem utilizada no SFIP

Uma imagem pode ser particionada em regiões com várias formas: triangulares, retangulares, quadradas, poligonais. A figura 3.2 exemplifica alguns tipos de partições. Tradicionalmente o particionamento em regiões quadradas tem sido o mais utilizado em codificações fractais. Uma partição triangular tem vantagens em relação as outras, pois pode se adequar a característica de uma determinada região da imagem, e não necessita ter seus lados paralelos as bordas da imagem, como na maioria das partições quadradas.

O motivo para a disseminação da partição quadrada deve-se ao fato de seu custo computacional ser baixo em relação às outras formas de regiões. Mas, novas formas de particionamentos vem sendo desenvolvidos e aprimorados, diminuindo assim o seu custo computacional.

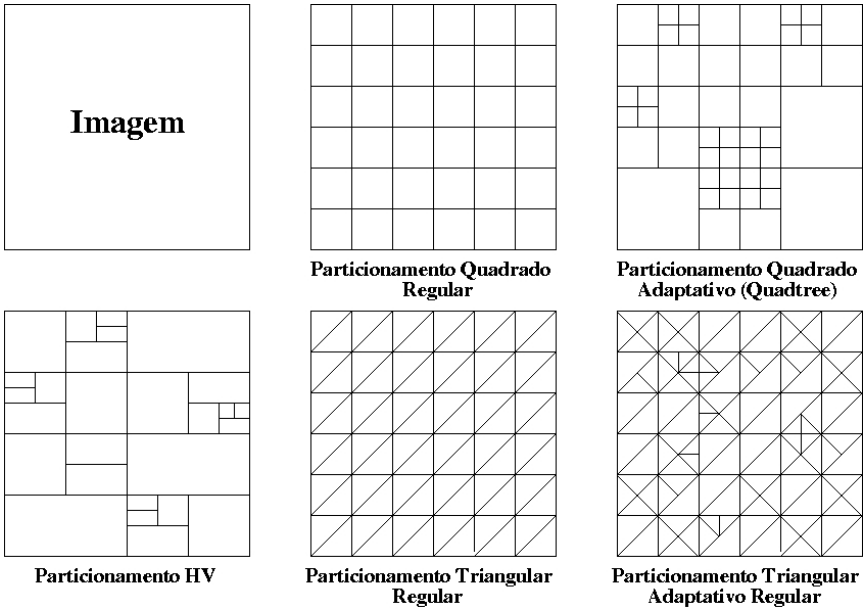


Figura 3.2 Diferentes formas de particionamento de uma imagem.

3.2 Considerações sobre a simetria do bloco domínio

Para cada bloco domínio tem-se 8 simetrias que serão testadas em relação ao bloco molde corrente, na determinação de m_i na equação (3.5). As simetrias são obtidas através da rotação e reflexão do bloco domínio reduzido, como mostrado na figura 3.3. Os blocos domínio reduzidos serão apresentados em detalhes no próximo item.

Para uma representação mais otimizada destas simetrias é possível considerar somente três operações, que são aplicadas sozinhas ou composição, no bloco domínio reduzido. As operações são:

- reflexão em relação ao eixo horizontal;
- reflexão em relação ao eixo vertical; e

- reflexão em relação a diagonal principal do bloco.

Como tem-se oito simetrias e três operações de simetria, três bits são suficientes para representá-las (2^3=8). Considera-se então o bit menos significativo como o responsável pela indicação de existência de reflexão horizontal, o bit intermediário como o responsável pela indicação de reflexão vertical, e finalmente o bit mais significativo como o responsável pela indicação de reflexão sobre a diagonal principal do bloco. Utilizando esta convenção tem-se, como apresentado na figura 3.3, as seguintes possibilidades:

(0,0,0) - Operação identidade

(0,0,1) - Reflexão em relação ao eixo horizontal

(0,1,0) - Reflexão em relação ao eixo vertical

(1,0,0) - Reflexão em relação à diagonal principal

(0,1,1) - Reflexão horizontal composta com reflexão vertical (rotação de 180°)

(1,0,1) - Reflexão horizontal composta com reflexão sobre a diagonal principal (ou rotação de 90°)

(1,1,0) - Reflexão vertical composta com reflexão sobre diagonal principal (ou rotação de 270°)

(1,1,1) - Reflexão horizontal composta com reflexão vertical e reflexão sobre a diagonal.

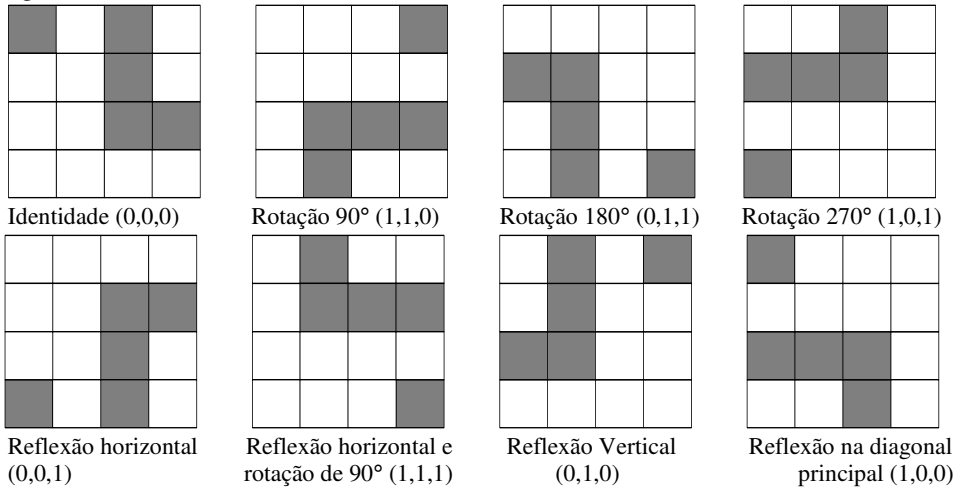


Figura 3.3 Bloco molde e suas 8 possíveis simetrias.

3.3 Procedimento básico de uma compressão automática

O procedimento básico segue uma implementação feita na referência [3], disponibilizada na Internet. Esta implementação será tratada neste trabalho como sendo o **programa base** a partir do qual serão introduzidas modificações.

Informações básicas:

Na compressão fractal automática a imagem é dividida em blocos de várias dimensões, e é necessário saber, para cada um destes blocos, a sua posição absoluta dentro da imagem.

Neste sentido, fica convencionado que a posição absoluta do bloco na imagem é a posição de seu primeiro pixel (mais à direita e acima) como exposto na figura 3.4:

Os pixels da imagem são contados no sentido de cima para baixo e da esquerda para a direita, desta forma, o pixel mais acima e a esquerda da imagem é o primeiro pixel do fluxo de dados da imagem (vetor imagem). Expandindo esta convenção, o primeiro pixel do fluxo do bloco segue a mesma orientação do vetor imagem. O índice do pixel do vetor imagem que corresponder ao primeiro pixel do vetor bloco indicará a posição deste bloco dentro da imagem.

Etapas da compressão fractal automática de uma imagem

A compressão fractal automática pode ser divida nas cinco etapas descritas a seguir.

Primeira etapa: (Definição dos blocos molde ou blocos range)

Inicialmente divide-se a imagem a ser comprimida em uma malha de blocos quadrados homogêneos de dimensões n x n pixels (por exemplo 4x4 pixels). Estes blocos não podem se sobrepor, e são chamados "range blocks"[3,10]. A figura 3.5 ilustra esta divisão. Nela tem-se uma imagem de 32x32 pixels dividida em blocos molde de 4x4 pixels, dando um total de $32/4x32/4=8^2=64$ blocos molde. Se esta imagem fosse de 512x512 pixels, teríamos $512/4x512/4=128^2=16.384$ blocos molde. O número de blocos molde depende do tamanho da imagem e do tamanho dos blocos molde, sendo em uma imagem de MxN pixels e blocos molde com dimensões fixas de n x n pixels igual à (M/n).(N/n).

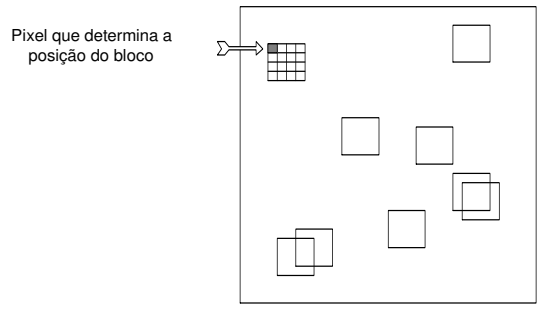


Figura 3.4 Posição dos blocos na imagem.

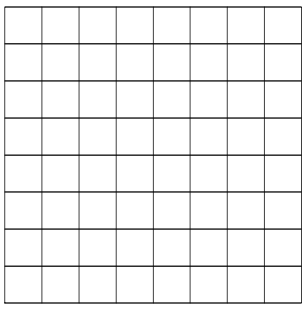


Figura 3.5 Malha de blocos molde.

Segunda etapa: (Definição dos blocos domínio)

A mesma imagem é, em uma segunda etapa, novamente dividida em blocos quadrados com o dobro da dimensão dos blocos imagem, 2nx2n (por exemplo 8x8 pixels). Estes blocos maiores podem se sobrepor, sendo denominados "blocos dominio". A figura 3.6 mostra estes blocos. Nela, a imagem de 32x32 pixels é dividida em $(32-8+1)^2=289$ blocos. Uma imagem de 512x512 pixels teria$(512-8+1)^2=255.025$ blocos domínio. O número de blocos domínio de uma imagem (MxN pixels) é igual à (M-2n+1).(N-2n+1).

Terceira etapa: (Redução dos blocos domínio)

Para tornar possível uma comparação entre os dois tipos de blocos criados nas etapas anteriores, os blocos domínio de 2n x 2n pixels terão as suas dimensões reduzidas à metade, passando a ter, então, n x n pixels. A redução processa-se tomando a média da

intensidade de sub-blocos de 2x2 pixels e atribuindo o valor da média quantizada ao pixel corresponde no bloco reduzido.

Quarta etapa: (Busca do melhor par domínio reduzido-molde)

Esta etapa consiste na busca do bloco domínio reduzido mais próximo, na métrica considerada, de um bloco molde (figura 3.7). Toma-se o primeiro bloco molde da imagem e utilizando uma métrica qualquer, por exemplo o erro rms [18], calcula-se a distância entre este bloco e todos os blocos domínio reduzidos e suas simetrias. Antes que se processe o cálculo da distância entre o bloco molde e o bloco domínio reduzido, este último ainda sofre um ajuste no offset (fator de deslocamento de intensidade - o) que é obtido através da diferença entre as médias de intensidades dos dois blocos anteriores (domínio reduzido e molde).

Temos 8 simetrias para cada bloco, e portanto serão feitos 8 testes para cada bloco domínio reduzido. Considerando por exemplo uma imagem de 32x32 pixels teremos 8x289=2.312 testes à realizar para cada bloco molde e 64x2.312=147.968 para a imagem toda, e se a imagem tiver 512x512 pixels teremos 8x255.025=2.040.200 para cada bloco molde, e 16.384x2.040.200=33.426.636.800 testes à realizar para toda a imagem.

São armazenadas a posição e a simetria (m_i) do bloco domínio que possuir a menor destas distâncias. O valor do fator de deslocamento (o_i) também será armazenado, e cada bloco molde passa a ser representado por uma tupla como:

$$W_i(D_x , D_y , m , o)$$

onde:D_x - determina a posição ao longo do eixo x do bloco domínio reduzido;
D_y - determina a posição ao longo do eixo y do bloco domínio reduzido;
m - define a simetria aplicada ao bloco domínio reduzido; e
o - define o deslocamento ao longo do eixo z aplicado no bloco domínio reduzido.

As contrações tonais e espaciais são definidas como constantes neste algoritmo. A contração espacial está ligada a razão entre o tamanho dos blocos imagem (n) e domínio (2n), ou seja, é considerada ½ ; e a contração tonal é considerada como um valor constante arbitrário, como por exemplo 3/4.

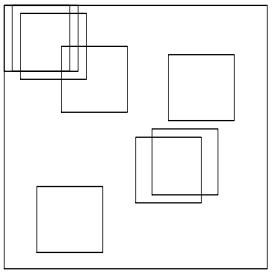


Figura 3.6 Blocos domínio.

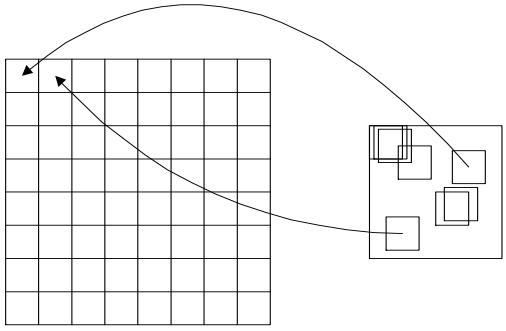


Figura 3.7: Formação dos pares de blocos ótimos.

Quinta etapa:

Repete-se a etapa anterior para cada um dos blocos molde. Armazenando-se as tuplas de cada bloco molde da imagem. O conjunto de todas estas tuplas será o código SFIP da imagem inteira.

3.4 Procedimentos otimizados

O procedimento básico descrito na seção anterior representa a forma originalmente proposta para uma compressão automática, conhecida como "busca exaustiva" ou "força bruta"{3, 10,11]. É um procedimento muito custoso e lento pois testa todos os blocos domínio reduzidos para cada bloco molde, sem considerar as características de cada bloco molde. Um bloco molde que possua, por exemplo, uma borda não precisaria ter em seu conjunto de blocos domínios reduzidos testados ("domain pool") aqueles que não possuíssem descontinuidades, ou altas freqüencias, características de bordas.

Várias técnicas de aceleração deste procedimento automático foram e estão sendo desenvolvidas. Técnicas que consideram as características de cada bloco molde, ou consideram propriedades de um dado conjunto de imagens etc. A seguir serão apresentadas algumas destas técnicas de aceleração do procedimento automático. Várias destas técnicas foram implementadas, para possibilitar sua comparação, num mesmo programa base, de modo a ser possível analisar suas características específicas em relação ao tempo de compressão e qualidade final da imagem processada [1]. A utilização do SFIP na representação da equação (3.2) considera o tamanho do arquivo comprimido como uma função do número de pixels da imagem (MxN) e do tamanho dos blocos molde (n x n). Nos exemplos que se seguem serão utilizadas imagens com o mesmo número de pixels e blocos molde e domínio com dimensões fixas, de modo que o tamanho do arquivo permanecerá constante em todas as técnicas. Para tornar clara as diferenças entre as imagens obtidas pela descompressão dos códigos de cada técnica utilizaremos sempre as mesmas imagens de teste.

Na compressão com perda temos 3 fatores intimamente relacionados, que são: (1) a velocidade do processo; (2) a qualidade da imagem restaurada; e (3) a taxa de compressão da imagem. As técnicas de aceleração do procedimento automático consideradas podem ser divididas em dois grupos (a denominação e a divisão que seguem é proposta nossa, buscam descrever as características dos métodos, não há ainda uma nomenclatura formal):

1. Técnicas de aceleração da busca do melhor par de blocos domínio-molde através da **sub-amostragem dos blocos domínio reduzidos**;
2. Técnicas que utilizam **categorização dos blocos domínio reduzidos** com o auxilio de uma estrutura de dados de armazenamento e busca.

Estes dois grupos básicos podem ser, também, subdivididos em alguns métodos específicos. Tem-se, então, para a primeira técnica os seguintes métodos:

1.1. Método de Busca Exaustiva Leve com passo Δ;
1.2. Método de Busca Local;
1.3. Método de Sub Busca Local com passo Δ;
1.4. Método de Busca Unitária;
1.5. Método de Busca em Área Restrita;
1.6. Método de Busca em Espiral Local.

A segunda técnica pode ser considerada com as seguintes subdivisões:

2.1. Armazenamento em estrutura de dados unidimensional, que, por sua vez, se divide nos métodos:

2.1.1 Método de classificação canônica através da média das intensidades dos quadrantes;

2.1.2 Método de classificação canônica através da média das variâncias dos quadrantes;

2.1.3 Método de dupla classificação canônica, através da média das intensidades e variâncias;

2.1.4 Método de classificação através da detecção de bordas pelo filtro de Sobel;

2.1.5 Método de criação de agrupamentos;

2.2. Armazenamento em estrutura de dados multidimensional, onde o armazenamento dos blocos domínio reduzidos é feito em uma árvore retangular multidimensional.

O programa base [3, 10] utiliza o procedimento básico de "Busca Exaustiva" descrito no item anterior. Ele adota blocos molde de 4 x 4 pixels e blocos domínio de 8 x 8 pixels. Todas as modificações implementadas no programa base seguirão estas especificações, sendo que qualquer derivação será notificada.

Os resultados obtidos pelos métodos implementados serão apresentados em imagens e tabelas como na figura 3.8. Primeiro a **imagem original** em teste é apresentada para facilitar a visualização das diferenças entre estas e a imagem **reconstruída** (segunda imagem). A terceira a imagem representa a diferença em módulo entre as duas primeiras pixel a pixel, (ou seja os **erros absolutos**) multiplicados por 5 para facilitará a visualização do erro, entre as imagens original e reconstruída, A quarta coluna também expõem os erros entre as imagens, mas desta vez o erro tratado pela equação:

$$(5 * \textit{ruído} / 2) + 127$$

onde: *ruído* = diferença entre os pixels correspondentes:imagem original - reconstruída. As constantes usadas nesta equação visam garantir que todos os valores obtidos para os erros sejam positivos mas que os sinais do *ruído* seja representado na imagem. Esta visualização das diferenças será chamada de **erro relativo deslocado**, e nelas cores mais claras que o cinza de fundo (erro=zero), indica, erros positivos, enquanto que tons de cinza mais escuros, indicam erros negativos.

Para comparações quantitativas entre a imagem original e a imagem reconstruída (obtida pela descompressão do código obtido pelos respectivos algoritmos) necessita-se de avaliações numéricas do processo de compressão. Essas são apresentadas como tabelas. A primeira destas é o **tempo** de codificação do algoritmo em relação ao mais rápido dos métodos fractais testado: Busca Unitária (que gasta 0,2 s em uma maquina Pentium de 133 MHz com 16MBytes de RAM e 256 de cache). Na referênica [1] estes resultados estão disponíveis em segundos (o que os torna dependentes do hardware usado). As demais funções de avaliação utilizadas são a **Raiz quadrada do Quadrado da Média dos Erros** (Root Mean Square Error - e_{rms}), a **Relação Sinal Ruído** (*rms*) (Signal to Noise Ratio - *SNR*), e a **Relação Sinal Ruído de Pico** (Peak Signal to Noise Ratio - *PSNR*), em decibéis.

Na função e_{rms} , dadas duas imagens de tamanho MxN pixels, uma imagem original $F(x, y)$ e uma imagem reconstruída $G(x, y)$, tem-se que o erro $e(x, y)$ entre elas para qualquer pixel (x, y) é dado por :

$$e(x, y) = G(x, y) - F(x, y) \qquad (3.6)$$

então, a raiz quadrada do quadrado da média dos erros, sobre todos os pixels da imagem (MxN) fica definida como:

$$e_{rms} = \sqrt{\left[\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[G(x,y)-F(x,y)\right]^2\right]} \tag{3.7}$$

A Relação Sinal Ruído (ms - mean square, média dos quadrados) á dada por:

$$SNR_{ms} = \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}G(x,y)^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}e(x,y)^2} = \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}G(x,y)^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[G(x,y)-F(x,y)\right]^2} \tag{3.8}$$

e a Relação Sinal Ruído *rms* é [18]:

$$SNR_{rms} = \sqrt{\frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}G(x,y)^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}e(x,y)^2}} = \sqrt{\frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}G(x,y)^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[G(x,y)-F(x,y)\right]^2}} \tag{3.9}$$

A relação Sinal Ruído de Pico, em decibéis (dB), é dada por:

$$PSNR = 20\log_{10}\left(\frac{2^n-1}{e_{rms}}\right) \tag{3.10}$$

como *n* é uma constante definida pela quantidade de bits necessária para representar cada pixel da imagem, no caso igual a 8, pois utiliza-se imagens em tons de cinza variando de 0 a 255, tem-se que a relação toma a seguinte forma:

$$PSNR = 20\log_{10}\left(\frac{255}{e_{rms}}\right) \tag{3.11}$$

Através destas quantificações é possível comparar os méritos relativos as várias implementações selecionando qual delas é mais adequada. Observa-se que a Relação Sinal Ruído de Pico (*PSNR*) é uma medida razoável da qualidade visual das imagens (dadas duas imagens reconstruídas, codificadas por diferentes métodos, a que apresentar o maior valor *PSNR* parecerá visualmente melhor). É claro, também, que a melhor codificação deverá ter a máxima compressão e a máxima fidelidade no menor tempo possível [9,18].

3.5 Resultado da implementação base

A figura 3.8 mostra os resultados obtidos com o programa base de Barnsley e Hurd [3] (sem qualquer tipo de modificação) para a imagem Lena (testes com outras imagens podem ser vistos em [1] ou no site: http://www.ic.uff.br/~felipe/imgtest.html). A taxa de compressão das imagens testadas com esse algoritmo (e métodos baseados nele) é constante, pois manteve-se o tamanho das imagens e uma malha de blocos-molde fixa (não adaptativa). Como as imagens de teste possuem dimensões de 128x128 pixels, tem-se em cada uma 16.384 pixels. O arquivo codificado gerado pelo programa base tem 7.172 bytes (57.376 bits). Desta forma, a taxa de compressão obtida pelo programa base será de 3, 5020 bpp. Essa método oferece uma imagem de boa qualidade e alta taxa de compressão, mas gasta um tempo 2 mil vezes maior que o método mais rápido[1].

Lena.tga | Reconstrução | Erro absoluto | Erro relativo

Tempo relativo	Erro rms	SNR	PSNR (dB)
2163, 15620	7, 6167	13, 3478	30, 4954

Figura 3.8 Resultados obtidos com o método base.

3.6 Técnicas de sub-amostragem dos blocos domínio reduzidos.

Na seção 4 serão apresentados os resultados obtidos com a introdução destas técnicas no programa base.

Método de busca exaustiva leve com Passo Δ

Neste procedimento deixa-se de testar todos os blocos domínios reduzidos. Toma-se todos os blocos domínio que estejam em posições múltiplas inteiras de Δ (figura 3.9). Por exemplo se Δ = 4, o conjunto de blocos domínio reduzidos (domain pool) é diminuído por um fator igual $1/\Delta^2 = 1/16$, diminuindo assim o tempo de busca. Como nem todos os blocos são pesquisados, o par domínio-molde ótimo pode ser perdido, diminuindo a qualidade da imagem atratora em relação a uma busca exaustiva. A degradação da imagem atratora aumenta significativamente quando a fração (fator $1/\Delta^2$) de blocos domínio utilizados no processo cai abaixo de aproximadamente 0.1.

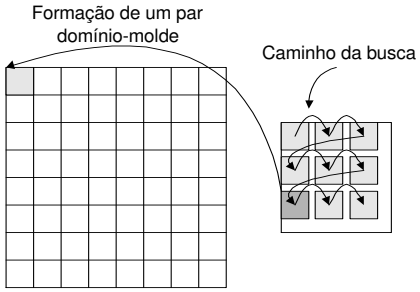


Figura 3.9 Formação do par domínio-molde no método de Busca Exaustiva Leve com passo Δ = 10.

Considerações sobre o tamanho das dimensões do bloco molde

Em uma codificação não adaptativa os blocos molde manterão sempre as mesmas dimensões. Estas deverão ter, então, um tamanho que permita que todas as freqüencias contidas no bloco molde (amostra) sejam completamente representáveis, e não haja qualquer perda. A efeito de perda por sub-amostragem é conhecido como *aliasing*.

Experimentalmente observa-se que uma imagem de dimensões 256x256 pixels é bem amostrada por blocos molde de dimensões 4x4 pixels, blocos molde maiores (8x8 pixels) provocam o aparecimento de *aliasing* (figura 3.10). Se estivermos trabalhando com

imagens de 128x128 pixels e blocos molde de 4x4 pixels teremos esse efeito indesejado. Para que ele não apareça deve-se utilizar blocos molde de dimensões 2x2 pixels perdendo efetivamente a grande taxa de compressão presente em uma codificação fractal; pois, tería-se quatro pixels sendo representados por 4 parâmetros, ou seja, uma compressão de aproximadamente 8 bpp; ou melhor, uma outra forma de representação da imagem, pois não haverían efetivamente uma compressão. Seguindo esta linha, uma imagem de 512x512 pixels já será bem amostrada com blocos molde de 8x8 pixels.

Em uma codificação adaptativa, que utiliza quadtree, o bloco molde poderá ter tamanhos variados, de acordo com as freqüencias que ele carregue. Um bloco com baixas freqüencias, tons de cinza quase constantes, poderá ter dimensões maiores como 32x32 pixels; enquanto que blocos molde com altas freqüencias, grandes variações nos tons de cinza, deverão ter dimensões menores, no mínimo, 4x4 pixels; para que haja efetivamente uma compressão em cada bloco molde da imagem.

Blocos molde de 2 x 2 pixeis

Blocos molde de 4 x 4 pixeis.

Lena (Blocos Molde de 2 x 2 pixels)			**Lena (Blocos Molde de 4 x 4 pixels)**		
Tempo Rel.	**Comp. (bpp)**	**PSNR (dB)**	**Tempo Rel.**	**Comp. (bpp)**	**PSNR (dB)**
27, 2396	6, 2354	34, 3935	1, 0138	1, 4282	27, 5964

Figura 3.10 *Aliasing* presente em uma compressão com blocos molde de 4x4 pixels em uma imagem de 128x128 pixels (obtidos com a implementação Dupla Classificação Canônica [7]).

Método de Busca Local

Neste método a pesquisa será restringida aos 80 blocos domínio reduzidos vizinhos ao bloco range corrente (figura 3.11). Considerando que uma boa parcela das imagens possuem a característica de ter regiões com variações contínuas de tonalidades, o resultado deste método é bastante interessante.

Método de Sub Busca Local com passo Δ

Este método é uma combinação dos dois anteriores. Neste, primeiro fazemos uma pesquisa nos blocos domínio reduzidos que estejam em posições múltiplas inteiras de um Δ, e então com o conhecimento da posição com o menor erro relativo, fazemos uma busca local nesta posição testando os 80 blocos vizinhos.

Método de Busca Unitária

Este é o método mais veloz deste tipo de compressão. Nele, o bloco domínio reduzido considerado no par domínio-molde é o que esta localizado na mesma posição do bloco

molde (figura 3.12). Como a posição do bloco domínio reduzido é implicitamente conhecida, existirá um ganho no tempo e, também, na taxa de compressão, pois teremos a tupla característica de cada SFIP com a seguinte aparência: W_i (m,o). Ou seja, a taxa de compressão será a maior de todos os métodos apresentados, usando blocos molde de mesmas dimensões. Neste caso a taxa de compressão em todas as imagens continua a ser função, apenas, das dimensões da imagem, MxN pixels, e das dimensões dos blocos molde.

Método de Busca em Área Restrita

Neste método a pesquisa é restringida à áreas próximas do bloco molde. Pode-se então restringir esta busca, por exemplo, ao quadrante da imagem onde está posicionado o bloco molde (figura 3.13). Desta forma, o tempo de busca será reduzido por um fator de ¼ em relação ao procedimento básico. É importante salientar que a busca em área restrita obtém melhores resultados quando a imagem possui característica locais semelhantes, áreas de suavidade ou áreas com variações de intensidades semelhantes.

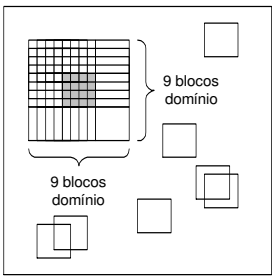


Figura 3.11 Blocos pesquisados no método de Busca Local.

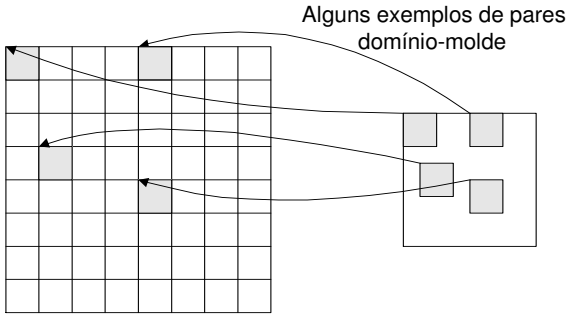


Figura 3.12 Par domínio-molde usado no método de Busca Unitária.

Consideração sobre a obtenção do melhor par domínio-molde

Existem várias métricas que podem ser utilizadas em uma codificação fractal, mas para cada tipo de implementação existe sempre uma, ou algumas que se adequam melhor que as outras. Nos métodos apresentados acima a métrica utilizada não necessitava ter como resultado valores bem comportados em um intervalo bem definido, pois todos os blocos de uma determinada região da imagem eram testados, sem exceção. A métrica neste caso tem que possuir um baixo custo computacional, proporcionando assim o aumento da velocidade da codificação. Por exemplo, considerando um bloco domínio reduzido "*D"* e um bloco molde "*M"*, ambos de n x n pixels, uma métrica de baixo custo computacional seria:

$$d(D(x,y), M(x,y)) = \sum_{x=0}^{n-1}\sum_{y=0}^{n-1}\left[D(x,y) - M(x,y)\right]^2 \tag{3.12}$$

Quando muda-se o tipo de enfoque do problema é importante passar a utilizar métricas melhor comportadas, que ofereçam valores dentro de intervalos bem delimitados. É o caso do próximo método e da maioria dos métodos de categorização dos blocos. Nestes métodos não procura-se o melhor par domínio-molde; mas, sim, um par que atenda a

determinadas condições, e que possua um erro dentro de uma certa tolerância. Um exemplo de métrica mais comportada seria o seguinte:

$$d(D(x,y),M(x,y)) = \sqrt{\frac{1}{n^2}\sum_{x=0}^{n-1}\sum_{y=0}^{n-1}\left[D(x,y) - M(x,y)\right]^2} \tag{3.13}$$

Método de Busca em Espiral Local

Este método considera um caminho espiral partindo da posição do bloco range corrente (figura 3.14). A espiral é percorrida tomando-se variações de um só pixel. Diferentemente dos métodos anteriores que examinam todos os blocos domínio reduzidos candidatos, para encontrar o melhor par, a busca termina assim que um par suficientemente bom seja encontrado. Esta busca pode ser extremamente rápida, apesar de haver alguma perda da qualidade da imagem.

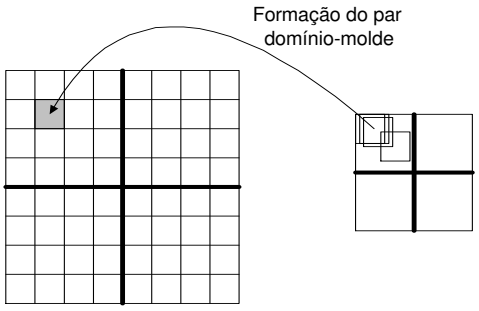


Figura 3.13 Método de Busca em Área Restrita.

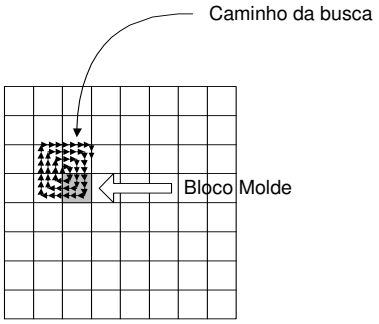


Figura 3.14 Busca em espiral.

3.7 Técnicas que utilizam categorização dos blocos domínio reduzidos com o auxilio de uma estrutura de dados de armazenamento e busca.

Uma forma diferente de aceleração é a obtida pelo agrupamento dos blocos domínio. Em geral, a classificação acelera a codificação particionando o conjunto de blocos domínio em um determinado número de classes *c*, baseando-se em propriedades específicas encontradas nos blocos. Desta forma, somente os blocos domínio da mesma classe que o bloco molde em questão seriam pesquisados, resultando em um fator de aceleração "c"; isto, sem considerar o tempo necessário para a classificação dos blocos domínio. Algumas das técnicas desta categoria subdividem cada bloco em 4 quadrantes. Como um bloco molde possui n x n pixels, cada quadrante terá n / 2 x n / 2 pixels.

Armazenamento em estrutura de dados unidimensional

Esta implementação (apresentada por Fisher [7]) difere do programa base em alguns aspectos. A primeira diferença refere-se à aplicação do fator de contração espacial, ½, na imagem. São geradas quatro imagens reduzidas distintas, ao invés de somente uma. A primeira das imagens reduzidas é obtida tirando-se a média de blocos de quatro pixels (2 x 2), que estejam posicionados em linhas ímpares e colunas ímpares. A segunda imagem reduzida é gerada através dos blocos posicionados em linhas ímpares e colunas pares. A

terceira, em linhas pares e colunas ímpares. A quarta, em linhas e colunas pares. Quando um bloco molde estiver posicionado em uma linha ímpar e coluna par, a busca do melhor bloco domínio para o par domínio-molde, será efetuada na imagem reduzida correspondente (linha ímpar e coluna par).

Outra diferença é a utilização do fator de contraste ***s***, juntamente com o fator de deslocamento ***o*** (offset) nas transformações afins dos SFIP. No programa base o fator de contraste não é utilizado, somente o fator de deslocamento. Isto implica em pequenas modificações nas funções de distância que são utilizadas na comparação entre os blocos domínio e molde. Estas funções, equação 3.14 , devem utilizar os valores da soma dos valores dos pixels do bloco molde e do bloco domínio reduzido, da soma do quadrado dos valores dos pixels do bloco molde e do domínio reduzido, e, ainda, da soma dos valores da multiplicação dos pixels correspondentes nos dois blocos (molde e domínio reduzido). Para tornar o programa mais otimizado, em velocidade, os valores da soma, e soma dos quadrados dos valores dos pixels tornam-se atributos dos blocos (passam a pertencer à estrutura definida, no código do programa, para o bloco domínio).

$$dis\tan cia = \sqrt{\frac{[rsum^2 + s(s.dsum^2 - 2.rdsum + 2.o.dsum) + o.(o.leng - 2.rsum)]}{leng}} \tag{3.14}$$

onde:s = fator de contraste;

o = fator de deslocamento;

leng = número de pixels do bloco molde;

rsum = soma dos valores dos pixels do bloco molde;

dsum = soma dos valores dos pixels do bloco domínio reduzido;

$rsum^2$ = soma dos valores ao quadrado dos pixels do bloco molde;

$dsum^2$ = soma dos valores ao quadrado dos pixels do bloco domínio reduzido;

rdsum = soma da multiplicação dos valores dos pixels correspondentes nos blocos domínio reduzido e molde;

Mais uma diferença ocorre quando o fator de contraste ***s*** obtido na melhor comparação entre os blocos domínio e molde é igual a zero. Neste caso, torna-se desnecessário o armazenamento da posição do melhor bloco domínio encontrado para o par domínio-molde. As demais diferenças entre os dois programas estão relacionadas à implementação do método. As classes do método da Dupla Classificação Canônica são preenchidas em uma primeira etapa do programa. Nem todos os blocos domínio são classificados. Somente os que estão no caminho equivalente ao caminho da Busca Exaustiva Leve com passo Δ são classificados. Desta forma, esta implementação torna-se mais veloz que o valor da aceleração obtida apresentado (8 x 72 = 576) no método Busca Exaustiva Leve com passo Δ [1,11]. Em uma segunda etapa os melhores pares domínio-molde são encontrados, de acordo com o método da Dupla Classificação Canônica [7].

Após a descompressão da imagem, a referência [7] adiciona uma etapa, de pós-processamento, onde é feita uma suavização da imagem através da aplicação de pesos aos pixels da imagem. Os pesos são selecionados de acordo com o nível da quadtree onde esteja o bloco domínio da transformação afim correspondente ao bloco molde em que esteja posicionado o pixel.

Método de classificação canônica pela média das intensidades dos quadrantes

Esta classificação canônica através da média das intensidades do quadrante é baseada na premissa de que os blocos domínio e imagem que possuam áreas similares claras e escuras tendem a possuir um erro de cobertura (colagem) menor que os outros. Para classificar um bloco, primeiro divide-se o bloco em quadrantes e calcula-se a intensidade média de cada um dos quadrantes. Estas intensidades são então ordenadas de 1 (mais clara) à 4 (mais escura), formando então um bloco de intensidades ordenadas de tamanho 2 x 2. Tem-se então 4! = 24 possibilidades de ordenações de intensidades.

Os blocos são classificados de acordo com o seu conteúdo, e o número da classe deve ser insensível aos ajustes de escala e offset aplicados aos blocos. Um bloco molde pode ser transformado por uma operação de simetria antes de ser comparado com um bloco domínio reduzido. Existem oito simetrias (4 rotações e 4 reflexões) e 24 classes, desta forma, as classes que podem ser permutadas por uma simetria são combinadas, formando 3 classes canônicas (figura 3.15). Quando um par é testado, a melhor simetria a ser aplicada é aquela que harmoniza as ordenações dos blocos. Esta simetria pode ser calculada imediatamente se soubermos a ordenação que leva cada um dos blocos à uma ordenação fixa (a ordenação canônica representada pelo primeiro bloco de cada classe). Se o bloco domínio reduzido e o bloco molde forem levados à orientação canônica através, respectivamente, das simetrias sim1 e sim2, então a simetria $sim1^{-1}$ o sim2 , apresentada na figura 3.16, levará o bloco molde à orientação do bloco domínio reduzido. As simetrias apresentadas na primeira linha da figura 3.15 transformam o bloco de forma a levá-lo a posição canônica de sua classe, representada pelo primeiro bloco de cada classe. As simetrias de um quadrado podem ser geradas por a e b, que representam respectivamente a rotação de 90^o no sentido anti-horário e a reflexão sobre uma linha média vertical.

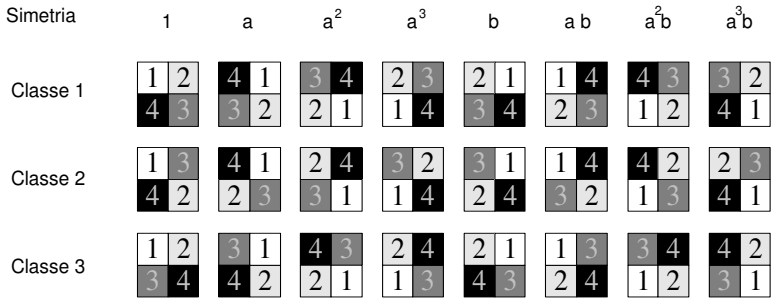


Figura 3.15 Possíveis orientações de um bloco em quadrantes.

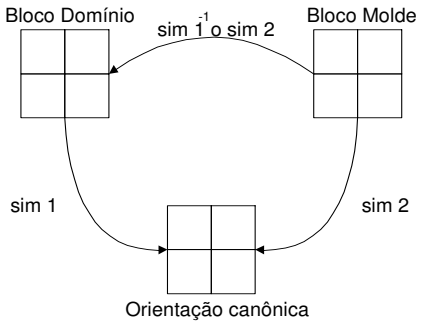


Figura 3.16 Relação entre as simetrias aplicadas aos blocos molde e domínio reduzido.

Então, para um dado bloco molde, a busca será efetuada somente sobre o conjunto de blocos domínio dentro da mesma classe, resultando em um fator de aceleração *c* igual a 3. Se armazenarmos a operação de simetria que leva o bloco domínio à orientação canônica de sua classe, pode-se eliminar o teste das oito simetrias de cada bloco. O fator de aceleração será, então, aumentado para c=3x8=24. Um fator de escala negativo aplicado a um bloco rearrumaria a ordem relativa dos quadrantes. Quando o sinal das médias dos quadrantes for modificado, a classe a que um bloco pertence não se modificará, somente a operação de simetria que o levará à posição canônica. No processo de busca, um bloco molde será comparado com duas classes, uma para fatores de escala positivos e outra para fatores negativos [1,7].

Uma implementação deste método pode ser obtida modificando-se algumas opções no programa desenvolvido por Fisher [7], disponibilizado na Internet (http://inls.ucsd.edu/y/Fractals), para a Dupla Classificação Canônica. Para os resultados mostrados na figura 3.17 e demias da referência [1], as opções do programa foram ajustadas para que ele utilize um particionamento não adaptativo (malha de blocos molde fixa) com blocos molde de dimensões 4 x 4 pixels, e continue a busca do melhor par domínio-molde em todas as sub classes (as sub classes correspondem a segunda classificação feita na dupla classificação canônica). As opções são as seguintes: m (min. part) = 5, M (max. Part) = 5, w (largura da imagem) = 128, h (altura da imagem) = 128, f = ativo (busca em todas as sub classes).

A taxa de compressão das imagens utilizadas varia, apesar de estar sendo utilizada uma malha de blocos molde fixa, pois quando o fator de escala aplicado à um bloco domínio reduzido é igual a zero, torna-se desnecessário o armazenamento da posição do bloco domínio. Na descompressão desta transformação, sem uma posição de bloco domínio definida, será utilizado o primeiro bloco domínio da imagem, posição 0.

Lena.tga Reconstrução Erro Erro

Tempo Relativo	Erro rms	SNR rms	PSNR (dB)	Comp. (bpp)
17, 5530	8, 2026	12, 4968	29, 8518	1, 4399

Figura 3.17 Resultados com o método de Classificação Canônica através das Intensidades.

Método de classificação canônica por média das variâncias dos quadrantes:

Neste método [13], como no anterior, as características dos blocos molde são consideradas e, com isso, muito cálculo é eliminado através da categorização dos blocos domínio antes de efetuar a comparação. Se um bloco molde possui uma borda forte (alta variação na tonalidade de cinza) de nada adiantará efetuar a busca sobre blocos que possuam variações suaves de tonalidades de cinza, será um esforço perdido.

A classificação da variância do quadrante é baseada na premissa similar de que os blocos domínio e molde que possuem áreas semelhantes de alta variância (contornos) e de baixa variância (áreas com tons de cinza aproximadamente contínuo) tendem a "casar-se" com um menor erro de cobertura que os outros. Para classificar o bloco, ele é dividido em quadrantes e é calculada a variância em cada quadrante. Esta é ordenada de 1 (maior) à 4 (menor), formando então um bloco de variância ordenada que, como no caso anterior possui 24 possibilidades. Estas 24 possibilidades são então mapeadas em uma das 3 formas canônicas.

Método de dupla classificação canônica

Este método é uma combinação dos dois métodos anteriores, através da média das intensidades dos quadrantes, e da média das variâncias dos quadrantes. Nele os blocos domínio reduzidos são primeiramente divididos em quatro quadrantes e, através da ordenação das intensidade de seus quadrantes, classificados de acordo com sua orientação canônica. Uma vez divididos nestas três classes maiores, os blocos sofrem uma nova ordenação de seus quadrantes. Agora, são ordenados da maior variância para a menor variância. Tem-se então 24 possibilidades em cada uma das 3 classes anteriores, alcançando um total de 72 classes. Faz-se o mesmo para cada bloco molde, e desta forma elimina-se a necessidade de testar as 8 simetrias pois existirá uma simetria que levará o bloco molde ao bloco domínio reduzido já classificado, conforme a figura 3.16.

Sem considerar o tempo necessário para esta categorização dos blocos, teremos uma aceleração, para esta classificação combinada, de aproximadamente 8x72=576, sendo 8 o fator obtido por eliminar do processo o teste de todas as simetrias do bloco domínio reduzido. Os resultados obtidos com a implementação de Fisher [7] são apresentados em [1] e na figura 3.18. Para estes resultados as opções do programa foram ajustadas para que utilize uma partição não adaptativa com blocos molde de dimensões 4x4 pixels. As opções são as seguintes : m (min. part)=5, M (max. Part)=5, w (largura da imagem)=128, h (altura da imagem)=128.

Lena.tga Reconstrução Erro Erro

Tempo Relativo	Erro rms	SNR rms	PSNR (dB)	Comp. (bpp)
1, 2581	10, 6345	9, 7362	27, 5964	1, 4282

Figura 3.18 Resultados obtidos com a Dupla Classificação Canônica.

Método de classificação através da detecção de bordas

O conjunto de blocos domínio reduzidos é classificado baseado no reconhecimento de forma (detecção de contornos) e múltiplos pontos de decisão (*tresholds*), usando a magnitude acumulada do gradiente dos blocos através dos operadores de Sobel, que fornece uma medida do contraste dos blocos junto com o grau de redundância. No caso de blocos molde de baixo contraste, uma sub-amostra dos pixels pode ser usada para comprimir o bloco molde. Isto aumenta a velocidade de compressão pois o número de cálculos é reduzido.

O algoritmo básico consiste em um esquema de particionamento com blocos de, no máximo, 2^6=64 pixels e, no mínimo, 2^2=4 pixels, com um espaçamento entre blocos domínio de 2 pixels. O processo de compressão envolve a convolução de cada bloco domínio com operadores vertical e horizontal de Sobel. Os gradientes combinados (∇f) fornecem uma medida única da força dos contornos e são acumulados para todos os contornos em cada bloco. Estes valores são então ordenados em ordem de magnitude de gradientes e divididos em N classes iguais, por exemplo N=4. Os pontos limites das classes, no exemplo, são os valores do gradiente acumulado nas posições .25, .5, .75 na lista ordenada de gradientes. Estes pontos limites dependem da imagem sendo comprimida. O custo computacional de ordenar os gradientes é mínimo quando é utilizada a rotina de quicksort [15]. Os gradientes são combinados como:

$$\nabla f = |G_x| + |G_y| \tag{3.5}$$

onde G_x e G_y são:

$$G_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Método de agrupamento

O conjunto de blocos domínio é dividido em agrupamentos, e à cada um destes agrupamentos é associado o seu representante, tendo cada um destes grupos a propriedade de ser pequeno o erro (erro rms) entre quaisquer blocos domínio. Para um dado bloco molde, a busca do melhor par com o bloco domínio é feita em 2 níveis. No primeiro nível o melhor agrupamento é localizado calculando-se somente a distância entre o bloco molde

e o representante do grupo. O bloco representante do grupo é o centróide deste grupo, que em geral não corresponde a nenhum bloco do grupo. No segundo nível, o melhor bloco domínio do grupo é localizado.

Classifica-se primeiro os blocos domínio, normalizados para ter média zero, em 3 classes pelo método de classificação da intensidade. Estes valores de intensidade não são modificados pois poderiam invalidar o teste de contração. O número de comparações é minimizado escolhendo-se o número de agrupamentos como sendo $\sqrt{nD}$, com cada grupo contendo $\sqrt{nD}$ blocos domínio, onde nD é o número de blocos domínio. É, então, importante ter um algoritmo de agrupamento capaz de gerar um número específico de grupos [1]. Nas referências [5,6] os blocos são agrupados com base nas suas dimensões fractais.

Armazenamento em estrutura de dados multidimensional

Este algoritmo faz o armazenamento dos blocos domínio reduzidos em árvore retangular multidimensional. Diferentemente dos apresentados no item 3.6, que conduzem sua busca pelos melhores pares entre os blocos fisicamente próximos a um bloco molde, faz a sua busca entre os blocos domínio estruturalmente vizinhos (semelhantes), garantindo assim a manutenção da qualidade da imagem comprimida, e para isto utiliza-se de uma estrutura de armazenamento e indexação multidimensional (r-tree , r*-tree,...) [17].

Na estrutura multidimensional os blocos são aninhados de uma maneira que mantém os blocos estruturalmente semelhantes próximos entre si. Os blocos semelhantes armazenados são então divididos em subconjuntos definidos pela menor área retangular que os englobe. Os blocos são tratados como sendo uma entidade simples, um vetor posição em um espaço de posições abstrato, onde cada ponto distinto representa um bloco diferente. A posição do vetor é derivada de uma ordenação linear dos valores das intensidades dos pixels dentro do bloco. Quando aplicada uma função de avaliação de distância a este espaço, a posição relativa entre dois vetores irá determinar o quão perto eles estão.

Este algoritmo exibe uma complexidade de tempo log-linear. Ou seja, com aumento das dimensões de uma imagem de MxN pixels para 2Mx2N pixels o tempo de compressão, que era t, passará a ser 2t. O tempo de compressão pode ser dividido em 3 estágios:

- Setup - blocos domínio candidatos com informação útil (ex. valor de variância) calculada antecipadamente são extraídos da imagem de entrada;
- Inserção - os blocos domínio são filtrados de acordo com os valores de variância e inseridos nas respectivas estruturas r-tree;
- Busca - as estruturas são utilizadas para encontrar o melhor par de blocos domínio-molde.

O estágio setup é uma função linear com relação ao aumento do tamanho da imagem, mas devido a filtragem dos conjuntos de blocos domínio, os tempos dos estágios de inserção e busca aumentam por menos de um fator de 4 quando a imagem quadruplica o seu tamanho [11].

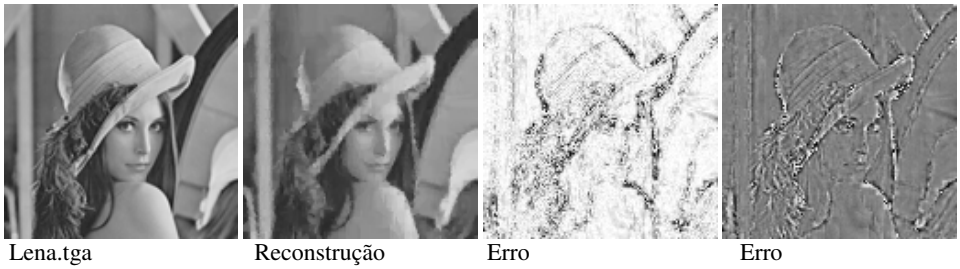
3.8 Codificação automática adaptativa

Uma forma de aumentar a taxa de compressão de uma codificação é a utilização de particionamentos adaptativos. Quando se tem uma região da imagem com baixas frequências, é possível utilizar partições maiores que englobem essas regiões; e quando ocorre o contrário, presença de altas frequências em uma determinada região da imagem, utiliza-se particionamentos menores, que possibilitem uma boa aproximação destas regiões.

Um bloco, independentemente de seu tamanho, será quase sempre representado por um mesmo número de parâmetros em uma codificação fractal. Fica fácil visualizar, que se for possível representar de forma satisfatória (dentro de uma certa tolerância) regiões cada vez maiores da imagem, será obtida uma taxa de compressão cada vez maior e mantido o nível de qualidade da imagem codificada [4].

Uma implementação bastante disseminada é a quadtree associada a particionamentos adaptativos. Nesta implementação, inicia-se com um particionamento grande, e dependendo da distância de colagem na formação do par domínio-molde, ser aceitável ou não, passa-se ou não um ao próximo nível na quadtree. Nesta implementação podem ser utilizados os dois tipos de quadtree, a completa e a parcial, que oferecem resultados distintos, sendo que a parcial costuma obter melhores taxas de compressão. Na quadtree parcial sub-regiões de uma região que já foi particionada somente serão novamente particionadas se não satisfizerem a tolerância especificada. Na quadtree completa, se uma das sub-regiões não satisfizer a condição definida, todas as sub-regiões, desta região específica, pertencentes ao mesmo nível da sub-região que não atende a condição, serão novamente particionadas mesmo que atendam a condição especificada.

Como exemplo do aumento da taxa de compressão de uma imagem através da utilização de particionamentos adaptativos, veja o resultado obtido com o programa de Fisher[7] com suas opções ajustadas para uma quadtree parcial de 3 níveis (-m 3, -M 5, -h 128, -w 128) (figura 3.19).



Lena.tga Reconstrução Erro Erro

Tempo Relativo	Erro rms	SNR rms	PSNR (dB)	Comp. (bpp)
1, 0046	10, 9677	9, 4636	27, 3285	1, 1162

Figura 3.19 Resultados com a Dupla Classificação Canônica com quadtree parcial de 3 níveis.

4. Resultados das Técnicas Implementadas

Esta seção apresenta os resultados obtidos com as técnicas implementadas no programa base de Busca Exaustiva [1,3]. Uma única imagem é apresentada nas figuras 4.1 a 4.9,

mas testes com outras (13 monocromáricas e 4 coloridas) foram executados e podem ser observados em: http://www.ic.uff.br/~felipe/mdfl.html. Os resultados das implementações possuem uma mesma taxa de compressão, pois é utilizada uma malha fixa, não adaptativa, de blocos molde. Nestas implementações foi utilizada uma subrotina de empacotamento de bits. Cada parâmetro de uma transformação da imagem codificada possui um certo número mínimo de bits necessários para representação de todas as possíveis variações deste parâmetro. Os parâmetros utilizados nestas implementações utilizam a seguinte quantidade de bits:

- posição x do bloco domínio reduzido ⇒ 6 bits;
- posição y do bloco domínio reduzido ⇒ 6 bits;

Obs: Como as imagens de teste têm dimensões de 128 x 128 pixels, e as imagens reduzidas possuem dimensões de 64 x 64 pixels (de onde sairão os valores das posições dos blocos domínio reduzidos), somente são necessários 6 bits para representar todo o intervalo, pois $2^6 = 64$.

- simetria considerada do bloco domínio reduzido ⇒ 3 bits;

Obs: Como os blocos possuem somente 8 simetrias, 3 bits é o suficiente para representá-las, pois $2^3 = 8$.

- offset ⇒ 8 bits.

Obs: Como podemos ter uma variação do offset de até 255, são necessários 8 bits para representá-la, pois $2^8 = 256$.

A quantidade de bits necessários à compressão de uma imagem é definida, então, pela quantidade de blocos molde multiplicada pela soma dos bits necessários a representação de cada um destes blocos. Como para uma imagem de 128x128 pixels existem 1024 blocos molde, serão necessários 1024x(6+6+3+8) bits, 23.552 bits ou 2.944 bytes. Uma imagem de 128 x 128 pixels possui 16.384 pixels, desta forma, a taxa de compressão para as imagens de teste será, então, de: 23.552 / 16.384 = 1,4375 bpp.

A taxa de compressão do método de Busca Unitária [1,11], devido as características do método já apresentadas na seção 3, é diferente da taxa de compressão dos outros métodos implementados. Neste método, os parâmetros das transformações utilizam as seguintes quantidades de bits:

- posição x e y do bloco domínio reduzido ⇒ 0 bit;

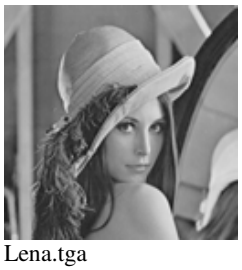
Obs: Como a posição do bloco domínio reduzido é implicitamente conhecida, não é necessário armazenar nenhum dos dois parâmetros de posição.

- simetria considerada do bloco domínio reduzido ⇒ 3 bits;

Obs: O número de simetrias continua sendo o mesmo.

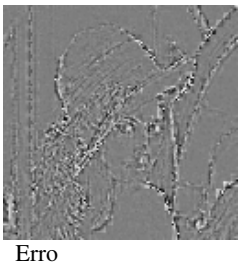
- offset ⇒ 8 bits.

Obs: A variação do offset continua dentro do mesmo intervalo.


Lena.tga


Reconstrução


Erro


Erro

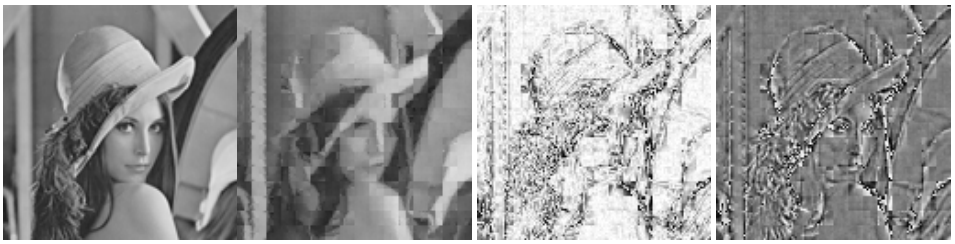
Tempo Relativo	Erro rms	SNR rms	PSNR (dB)
151, 1935	9, 7090	10, 5213	28, 3873

Figura 4.1 Busca Exaustiva Leve com passo Δ=4 e blocos molde de 4x4 pixels.

Lena.tga Reconstrução Erro Erro

Tempo Relativo	Erro rms	SNR rms	PSNR (dB)
39, 9032	11, 2420	9, 1418	27, 1139

Figura 4.2 Busca Exaustiva Leve com passo Δ=8 e blocos molde de 4x4 pixels.



Lena.tga Reconstrução Erro Erro

Tempo Relativo	Erro rms	SNR rms	PSNR (dB)
26, 8470	16, 5579	6, 3667	23, 7507

Figura 4.3 Busca Exaustiva Leve com passo Δ=4 e blocos molde de 8x8 pixels.

A quantidade de bits necessários à compressão de uma imagem será definida da seguinte forma:1.024x(3+8)=11.264 bits ou 1.408 bytes. A taxa de compressão para as imagens de teste é:11.264/16.384=0,6875 bpp.

Lena.tga Reconstrução Erro Erro

Tempo Relativo	Erro rms	SNR rms	PSNR (dB)
9, 1430	10, 5029	9, 5570	27, 7046

Figura 4.4 Resultados obtidos com Busca Local.

Lena.tga Reconstrução Erro Erro

Tempo Relativo	Erro rms	SNR rms	PSNR (dB)
41, 4430	9, 1887	11, 1621	28, 8657

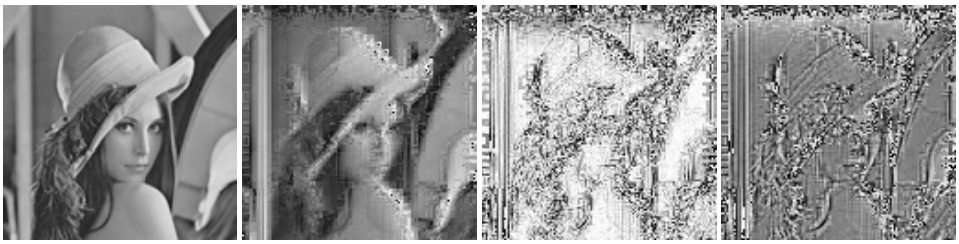
Figura 4.5 Resultados com Sub Busca Local com Δ=4 e blocos molde de 4x4 pixels.

Lena.tga Reconstrução Erro Erro

Tempo Relativo	Erro rms	SNR rms	PSNR (dB)
17, 6330	9, 9057	10, 3072	28, 2131

Figura 4.6 Resultados com Sub Busca Local com Δ=8 e blocos molde de 4x4 pixels.

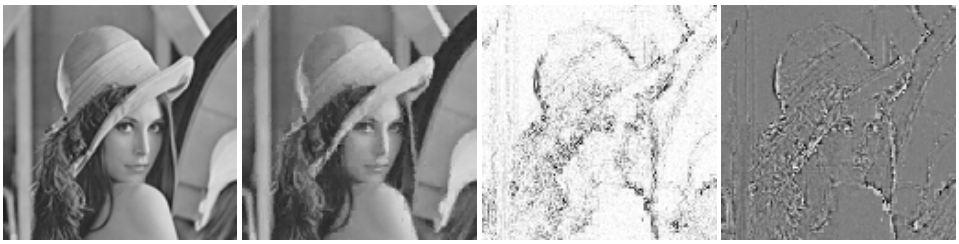
Como pode ser bem notado o método de Busca Exaustiva Leve com passo Δ = 4 e blocos molde de 8 x 8 pixels, resulta reconstrução com péssima qualidade devido a um efeito de sub-amostragem (devido ao tamanho das dimensões do bloco molde). Por este motivo, não será utilizado novamente o bloco molde com dimensões 8 x 8 pixels nas codificações não adaptativas deste trabalho.



Lena.tga Reconstrução Erro Erro

Tempo Relativo	Erro rms	SNR rms	PSNR (dB)
0, 2170	26, 3510	3, 8694	19, 7148

Figura 4.7 Resultados obtidos com o método de Busca Unitária



Lena.tga Reconstrução Erro Erro

Tempo Relativo	Erro rms	SNR rms	PSNR (dB)
106, 6520	8, 6244	11, 7174	29, 4162

Figura 4.8 Resultados obtidos com o método de Busca em Área Restrita.

Lena.tga Reconstrução Erro Erro

Tempo Relativo	Erro rms	SNR rms	PSNR (dB)
8, 6960	11, 2436	9, 1764	27, 1127

Figura 4.9 Busca em Área Restrita associado à Busca Exaustiva Leve com passo $\Delta = 4$.

5. Conclusões

Apresenta-se neste trabalho os fundamentos matemáticos que levam a possibilidade de utilizar a geometria fractal na compressão de imagens. A idéia básica de como automatizar a compressão de imagens utilizando o modelo fractal é discutida em detalhes, bem como algumas metodologias que buscam otimizar o tempo de codificação, que em última análise é o seu principal aspecto negativo. Suas principais vantagens são a taxa de compressão, o tempo extremamente rápido de descompressão, a facilidade de imclusão em métodos de busca de dados [16] e a sua simplicidade conceitual (se comparadas a outras técnicas de codificação com perdas semelhantes baseadas em transformadas de cosenos ou wavelets) [17]. Comparações mais detalhadas entre estes métodos considerando os diversos aspectos quantitativos que ajudam a avaliar a qualidade visual da imagem (Tempo de codificação, o Erro *rms*, a Relação Sina Ruído *rms* e a Relação Sinal Ruído de Pico) podem ser encontrada em http://www.ic.uff.br/~felipe/mdfl.html [1].

Abstract. With the sprouting of digital images, due to the growth of image databases and multimedia systems, and more intense use of nets, the necessities of efficient storage and transmission of images increase. Several new forms to transmit and store images efficiently have appeared. Fractal codification can be considered one of these new forms. This lecture has the objective to present methodologies of this type of codification, ranging from simple sets to complex

grayscale images. It compares several methodologies suggested in other works. The techniques: Light Brute Force, Local Sub Search, Local Search, Look Same Place Search, Restricted Area Search , and Local Fractal Dimension had been implemented in the same basic program of fractal compression by Brute Force. The techniques Double Canonic Classification, Canonic Classification through the Intensities, Double Canonic Classification with Quadtree of 3 levels had been analyzed from the program implemented by Fisher (http://inls.ucsd.edu/y/Frac). These comparisons consider visual quality, codification Time, the Error rms, the Signal to Noise Ratio rms and the Peak Signal to Noise Ratio.

Referências

[1] F.R. Aquino, "Um Estudo sobre Metodologias de Codificação Fractal", *Dissertação de Mestrado. CAA. UFF*, (1998) (http://www.caa.uff.br/~felipe).

[2] M.F. Barnsley, "Fractals Everywhere", 2nd Ed., Academic Press, New York, 1993.

[3] M. F. Barnsley, L. Hurd, "Fractal Image Compression", AK Peters, Wellesley, 1993.

[4] Chwen-Jye Sze , Hong-Yuan Mark Liao, Kuo-Chin Fan, Ming-Yang Chern, Chen-Kou Tsao, Fractal image coding system based on an adaptive side-coupling quadtree structures, *Image and Vision Computing*, 14 (1996) ,401-415.

[5] A. Conci, F.R. Aquino, Fractal image coding by multi-scale selection based on block complexity, *Journal for Geometry and Graphics*, Heldermann Verlag, Germany, ISSN 1433-8157 vol. 3 No. 1, (1999) pp. 57-65 (ou RT-02/99 em http://www.caa.uff.br).

[6] A. Conci, F.R. Aquino, Using adaptive contraction for fractal image coding based in local fractal dDimension, Proceedings SIBGRAPI, pp. 231-239 pub. IEEE Computer Science - ISBN 0-7695-0481-7/99, UNICAMPI-SP, 1999.

[7] Y. Fisher, "Fractal Image Compression: Theory and Application", Springer-Verlag, New York, 1995.

[8] C. Hart, Fractal image compression and recurrent iterated function systems, *IEEE Computer Graphics and Applications*, July, (1996), 25-40.

[9] E.W. Jacobs, Y. Fisher, R.D. Boss, Image compression: a study of the iterated transform method, *Signal Processing*, **29** (1992) 251-63.

[10] A. Jacquin, Image coding based on a fractal theory of iterated contractive image transformations, *IEEE Transactions on Image Processing*, **1**, No. 1, (1992) 18-30.

[11] J. Kominek, "Advances in Fractal Compression for Multimedia Applications", *Internal Report CS95-28*, (ftp://links.uwterloo.ca:/pub/Fractals/Papers/Waterloo kominek95c.ps.gz) University of Waterloo,(1995).

[12] C. Kubrusly, "Elements of Operator Theory", Birkhäuser, Boston, 2001.

[13] K. Lee, W. K. Lee, Fast fractal image block coding based on local variances, *IEEE Transactions on Image Processing*. **7**, No. 6, (1998) 888-891.

[14] L. Lu, *Fractal Imaging*, Academic Press, San Diego, 1997.

[15] S. Morgan, A. Bouridane, Application of shape recognition to fractal based image compression, *Proceedings* IWSIP, pp. 665-668, Elsevier Science, 1996.

[16] M. Nappi, G. Polese, G. Tortora, FIRST: Fractal Indexing and Retrieval SysTem for image databases, *Image and Vision Computing*, **16**, (1998) 1019-1031.

[17] A. Watt, F. Policarpo, "The Computer Image", ACM SIGGRAPH Series, Addison-Wesley, Essex, 1998.

[18] R. Weeks Jr. "Fundamentals of Electronics Image Processing", IEEE&SPIE Press., 1996.