Universidade Nova de Lisboa
Faculdade de Ciências e Tecnologia
Departamento de informática

Compressão de Imagens coloridas com fractais

Por
António Manuel Rodrigues Manso

Dissertação apresentada na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa para a obtenção do grau de mestre em Inteligência Artificial Aplicada.

Orientador : Prof. Doutor Luís Miguel Parreira e Correia

Lisboa
Fevereiro de 2002

À memória de meu pai.

Clouds are not spheres, mountains are not cones, coastlines are not circles, and bark is not smooth nor does lightning travel in a straight line.

(Benoit B. Mandelbrot)

Agradecimentos:

Não podemos deixar de agradecer a todos aqueles que, de diferentes maneiras, contribuíram para que esta dissertação chegasse a bom porto. O nosso reconhecimento vai, desde logo, para o meu orientador, Prof. Doutor Luís Miguel Parreira e Correia, pela sugestão do tema, pelo valioso acompanhamento técnico e científico e pela disponibilidade incondicional. Agradecemos, em seguida, à Cristina, à Lucy e ao Henrique, pelo seu contributo ao nível da elegância linguística do documento. Estendemos, como é óbvio, a nossa gratidão aos pais e aos professores, que nos incutiram o indescritível prazer de descobrir de investigar e de fazer. Não esquecemos, igualmente, os nossos colegas e amigos, pelas sugestões, observações, comentários e testes feitos durante a elaboração deste trabalho. Não podemos deixar de agradecer à Isabel, pela paciência e disponibilidade demonstradas durante o nosso "exílio" académico. Por fim, o nosso obrigado a todos aqueles que, de uma forma ou de outra, contribuíram para que este trabalho atingisse os objectivos a que nos propusemos.

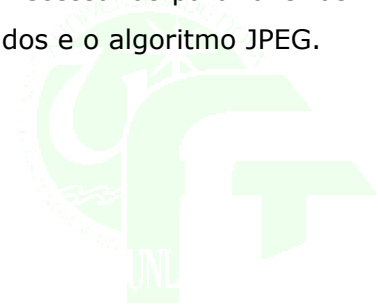
Sumário:

Este trabalho implementa um desenvolvimento da técnica de compressão de imagens com a tecnologia dos fractais do tipo IFS (Iterated Function System) aplicada a imagens coloridas, utilizando a técnica de Jacquin e tendo por base o algoritmo de Yuval Fisher que foi publicado em [FIS95].

O trabalho acima citado fornece dois métodos distintos de efectuar o trabalho a que nos propomos, abordando-os, porém, de uma forma ligeira e sem apresentar resultados práticos.

Além disso, a bibliografia consultada permitiu-nos descortinar dois métodos adicionais, que foram desenvolvidos de forma a comprimirem imagens coloridas com superior qualidade e/ou quantidade.

Temos, assim, um total de quatro métodos que permitem obter a compressão de imagens coloridas com a técnica fractal. A dissertação descreve os métodos de compressão adaptados e melhorados e apresenta uma análise crítica dos resultados práticos obtidos pelo protótipo de software que acompanha o presente trabalho. De modo a termos uma visão global da complexidade das imagens coloridas, a dissertação procede a uma breve incursão pelo sistema visual humano e pelos espaços de cor que suportam as imagens a serem tratadas, bem como as métricas necessárias para fazer as comparações entre os resultados obtidos pelos métodos explorados e o algoritmo JPEG.



Abstract

This paper develops the fractal image compression technique with fractal IFS type technology applied to colored images with the Jacquin technique based on the Yuval Fischer algorithm published in [FIS95].

The technique mentioned above gives two distinct methods of accomplishing the task we propose. However, we do not deal with them in depth, nor do we present practical results.

The consulted bibliography allowed us to reveal two other methods that were developed to compress colored images with superior quality and/ or quantity.

We therefore have a total of four methods that allow us to obtain the compression of colored images with the fractal technique. The dissertation describes the compression methods with adaptations and improvements and presents a critical analysis of the practical results obtained by the software prototype that accompanies the present paper.

In order to have a global vision of the complexity of the colored images, the dissertation gives a brief overview of the human visual system and the color spaces that support the images to be treated as also the necessary metrics to compare the results of the methods investigated and the JPEG algorithm.

Glossário

Atractor	Imagem final obtida por um IFS
CMY	Cian Magenta Yellow – Espaço de cor
CMYK	Cian Magenta Yellow Black – Espaço de cor
HSL	Hue Saturation Luminance – Espaço de cor
IFS	Iterated Function System – Sistema de funções iteradas, conjunto de transformações afim
JPEG	Joint Photographic Experts Group – Algoritmo de compressão de imagens
MMCR	Máquina de Múltiplas Cópias Reduzidas
MSE	Mean Square Error – medida de fidelidade entre imagens.
PIFS	Partioned Iterated Function System- IFS que funciona com regiões da imagem, ao contrário do IFS que funciona com a imagem toda
PSNR	Peek Signal Noise Ratio – medida de fidelidade entre imagens.
RGB	Red, Green, Blue – Espaço de cor
SNR	Signal Noise Ratio - Medida de fidelidade entre imagens.
Teorema da colagem	Teorema que suporta a compressão de imagens com fractais, desenvolvido por Barnslay.
Transformação afim	Sistema de equações na forma AX + T, que possibilitam, rodar, escalar e transladar o ponto X.
YCrCb	Luminância(Y) Crominância red (Cr) crominância blue (Cb) – Espaço de cor

Índice

Agradecimentos: 3
Sumário: 4
Abstract 5
Índice 7
Listas das Figuras 9
Listas das Tabelas 11
Listas das Equações 12
Introdução 14

CAPÍTULO 1: COMPRESSÃO DE IMAGENS COM FRACTAIS 17
Introdução 17
Solução do problema inverso - abordagem clássica 18
Pressupostos para a compressão de imagens através de um IFS 22
Abordagem clássica – Algoritmo de Barnslay 24
Abordagem contemporânea – Algoritmo Jacquin 26
Extensão para a escala de cinza 28
Etapas da compressão com fractais 31
Segmentação da imagem em regiões 31
Segmentação da imagem em domínios 35
Estratégias de procura 36
Descodificação 37
Considerações finais 38

CAPÍTULO 2: A COR 39
O sistema visual humano 39
Imagens digitais 41
Discretização e quantificação 41
Modelação da cor 42
Sistema XYZ 42
Sistema RGB 44
Sistema CMY e CMYK 45
Sistemas de Vídeo (YUV, YCrCb) 45
Sistema HSV 47
Sistemas enumerados 48
Representação de imagens nos diferentes espaços de cor 48
Representação em CMY e CMYK 49
Representação em YUV e YCrCb 49
Representação em HSL 52
Representação no sistema enumerado 52

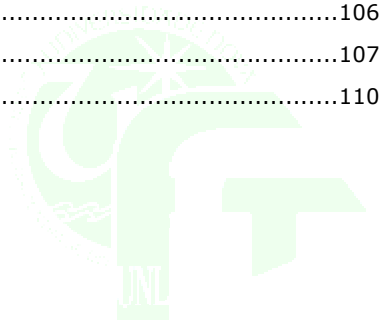
CAPÍTULO 3: COMPRESSÃO DE IMAGENS COLORIDAS COM FRACTAIS 53
Trabalhos publicados 53

Camadas separadas 53
Camadas Unidas 54
Camadas principal 56
Camadas Reduzidas 57
Compressão Híbrida 58
Partilha dos domínios 58
Métricas da compressão 59
Métrica PSNR para imagens coloridas 59
Taxa de compressão 61

CAPÍTULO 4: DEFINIÇÃO DOS TESTES PRÁTICOS 63
Escolha das Imagens 63
Parâmetros da compressão 64
Configurações pré-definidas 69
Domínios em cada configuração pré-definida 71
Comparação dos Resultados 72

CAPÍTULO 5: RESULTADOS PRÁTICOS DOS ALGORITMOS 74
Compressão de imagens através de camadas separadas 74
Domínios separados 76
Crítica dos resultados 79
União dos Domínios 80
Crítica dos resultados 81
Comparação entre domínios separados e domínios unidos 82
Compressão de imagens através de camadas dependentes 83
Crítica dos resultados 87
União dos domínios 89
Camada principal 89
Crítica aos resultados 93
União dos domínios 95
Camadas reduzidas 95
Crítica aos resultados 100
Algoritmo híbrido 100
Crítica aos resultados 102
Descompressão das imagens 102
Conclusão 103

CONCLUSÃO 105
Trabalho futuro 106
Bibliografia 107
Recursos na Internet 110



Listas das Figuras

figura 1.1 - Máquina de Múltiplas Cópias Reduzidas [KOM95a].....18
figura 1.2 – Imagem original e o resultado das operações permitidas pela MMCR [BRAN99] (adaptado).....18
figura 1.3 – Imagem original e as 5 primeiras iterações da MMCR da figura 1.1 aplicadas a um quadrado.....19
figura 1.4 - Triângulo de Serpinsky.....19
figura 1.5 – Aplicação da matriz A a um quadrado.....20
figura 1.6 – Aplicação de uma transformação afim IFS a uma figura [BAR88].....20
figura 1.7 – Primeiras três iterações obtidas através do IFS da equação 1.5 de duas imagens distintas.....21
figura 1.8 – Imagens obtidas através de IFS.....23
figura 1.9 – Imagem a comprimir.....25
figura 1.10- Imagem comprimida.....26
figura 1.11 – Região autosimilar com um domínio na imagem Lena.....27
figura 1.12 - Imagem que vai se submetida às transformações.....27
figura 1.13 - Representação gráfica do algoritmo de Jaquin.....28
figura 1.14 - Codificação de uma região.....28
figura 1.15 – Representação tridimensional da imagem Lena [FIS95].....30
figura 1.16 – Codificação de um região tridimensional [GIG97] (adaptado).....30
figura 1.17 –Partições fixa e dinâmica (Quadtree, Horizontal-Vertical e Regiões Irregulares).....33
figura 1.18 – Partições triangulares.....33
figura 1.19 – Partições poligonares.....34
figura 1.20 – Partição da imagem Lena com quatro métodos distintos [www 1] (adaptado).....34
figura 1.21 – Quadtree de 3 níveis com domínios do tipo D1 para uma imagem 256x256 com grelha de tamanho 16.....35
figura 1.22 - Quadtree de 3 níveis com domínios do tipo D2 para uma imagem 256x256 com grelha de tamanho 16.....36
figura 1.23 - Quadtree de 3 níveis com domínios do tipo D3 para uma imagem 256x256 com grelha de tamanho 16.....36
figura 1.24 – Classes principais [FIS95].....37
figura 1.25 – 5 Iterações da descodificação de imagem lena.....37
figura 2.1 - Comprimentos de onda da luz e o espectro visível [SCR99].....39
figura 2.2 - Sitema visual humano [ARE95] (adaptado).....40
figura 2.3 - Ampliação da Retina [ARE95] (adaptado).....40
figura 2.4 – Fracção da luz absorvida por cada tipo de cone para as frequências do espectro visível [APO98].....40

figura 2.5 - Imagem analógica. 42
figura 2.6 - Imagem digital. 42
figura 2.7 – Pirâmide do sistema XYZ e o espectro visível [www 2]. 43
figura 2.8 - Reconstrução de cor do sistema RGB no XYZ - dados experimentais de 1931 e 1964 [SCR99]. 43
figura 2.9 – Cores visíveis Sistema XYZ (projecção) [www 2]. 43
figura 2.10 – Cores disponíveis num monitor RGB típico [www 2]. 43
figura 2.11 – Cubo RGB [SCR99]. 44
figura 2.12 – Cores no Sistema HSL (tom, saturação, luminância). 47
figura 2.13 - Sistema de Munsell (1905). 48
figura 2.14 – Imagem da Lena com 24 bites de cor no sistema RGB. Componentes e respectivos histogramas. 49
figura 2.15 - Componentes CMY. 49
figura 2.16 – Componentes YUV e respectivos histogramas. 50
figura 2.17 - Componentes YCrCb e respectivos histogramas 51
figura 2.18 - Componentes HSL e respectivos histogramas. 52
figura 2.19 – Imagem da Lena no sistema enumerado com 256 cores. 52
figura 2.20 - Palete de cores da figura 2.19. 52
figura 3.1 – Esquema de compressão com camadas separadas. 54
figura 3.2 - Esquema de compressão com camadas dependentes. 55
figura 3.3 - Esquema de compressão com camada principal. 56
figura 3.4 – Esquema de compressão com layers reduzidos. 57
figura 3.5 - Esquema de compressão híbrido 58
figura 4.1 – Imagens utilizadas nos testes. Babbon, Peppers, Lena e Sky. 64
figura 4.2 - Parâmetros da compressão de imagens com fractais. 65
figura 5.1 – codificação de três regiões com o método de camadas separadas e domínios unidos. 81
figura 5.4 – Codificação de três regiões com o método de camadas separadas e domínios separados. 83
figura 5.10 – Compressão de uma região com o método de camadas dependentes e domínios unidos. 89

Listas das Tabelas

tabela 1.1– Conjunto de transformações afim (IFS) que comprimem a figura 1.13. 26
tabela 1.2– Tabela de transformações de Jaquin (rotação e escala).....28
tabela 3.1- Quatro imagens da Lena com diferentes valores de PSNR. 61
tabela 3.2- Quatro imagens da Lena com diferentes taxas de compressão.....62
tabela 4.1 – Variação do PSNR de várias imagens em relação ao valor máximo do factor de escala. 66
tabela 4.2 – Cabeçalho do ficheiro das imagem comprimida.....68
tabela 4.3 – Valores de identificação do tipo de compressão 68
tabela 4.4 – Codificação de uma transformação no ficheiro de imagem comprimida.....69
tabela 4.5 – Parâmetros de compressão com ênfase na fidelidade 70
tabela 4.6 - Parâmetros de compressão com ênfase na compressão.....71
tabela 4.7 - Número de domínios em cada nível da partição qadtree. 72
tabela 4.8 – Número de bites para a codificação de um domínio.....72
tabela 5.1 - Número de bites de uma transformação afim no método das camadas separadas sem a codificação do domínio. 75
tabela 5.2 - Resultados práticos do algoritmos de camadas separadas com domínios separados....76
tabela 5.3 – Resultados práticos do algoritmos de camadas separadas com domínios unidos.....81
tabela 5.4 - Número de bites de um transformação IFS no método das camadas separadas sem a codificação.....84
tabela 5.5 - Resultados práticos do algoritmos de camadas dependentes com domínios separados. 84
tabela 5.6 - Resultados práticos do algoritmos de camada principal com domínios separados. 90
tabela 5.7 - Resultados práticos do algoritmo de camadas reduzidas. 96
tabela 5.8 - Resultados práticos do algoritmo híbrido. 101
tabela 5.9 – As quatro primeiras iterações da descompressão da imagem Lena pelo método de descompressão modificado. 103

Listas das Equações

equação 1.1 - Transformação afim de um ponto no plano 2D 19
equação 1.2 - Matriz A (rotação, escala).....20
equação 1.3 - Transformação pontual do IFS composto pelas transformações afim w_1, w_2, ... w_n. ..20
equação 1.4 – Critério de convergência de uma tranformação afim w. 21
equação 1.5 - Conjunto de transformações afim , IFS , que geram o triângulo de Serpinsky. 21
equação 1.6 – Imagens obtidas através de um IFS.....21
equação 1.7- Imagem final do IFS.....22
equação 1.8 – Transformação afim com extensão à escala de cinza. 29
equação 1.9 – Obtenção do domínio transformado (Dt) através da simetria Sk 29
equação 1.10– Distância entre a região R_i e o domínio D_j.....29
equação 1.11 – Quadrado da diferença entre a região e o domínio 29
equação 1.12- Factor de escala para a intensidade do pixel 30
equação 1.13 - Deslocamento para a intensidade do pixel.....30
equação 1.14 – Deslocamento quando a escala for zero.....30
equação 1.15 - Distância entre o domínio e a região (métrica rms).....30
equação 2.1 - Conversão do espaço RGB para CMY 45
equação 2.2 - Conversão de CMY para CMYK 45
equação 2.3 - Cálculo da luminância através das componentes RGB.....46
equação 2.4 - Transformação de RGB para YUV.....46
equação 2.5 - Transformação de YUV para RGB.....46
equação 2.6 - Transformação de RGB para YCrCb. 47
equação 2.7 - Transformação de YCrCb para RGB. 47
equação 2.8 - Conversão de RGB - CMY - RGB. 49
equação 2.9 - Conversão RGB – YUV. 50
equação 2.10 – Conversão YUV – RGB. 50
equação 2.11 - Conversão RGB – YCrCb. 51
equação 2.12 - Conversão YCrCb – RGB. 51
equação 3.1 – Tranformação afim com 5 dimensões. 54
equação 3.2 – Quadrado da diferença entre um pixel. 59
equação 3.3 – MSE – (Erro quadrático médio).....59
equação 3.4 – SNR – (Rácio entre o sinal e o ruído). 59
equação 3.5 - SNR em escala logarítmica. 60

equação 3.6 – PSNR – (Razão entre o sinal e o sinal de pico). 60
equação 3.7 – MSE para imagens coloridas. 60
equação 3.8 – PSNR para imgens coloridas. 60
equação 3.9 – PSNR para imagens com 255 níveis em cada componente.....60
equação 3.3.10 – Cálculo do rácio de compressão 61
equação 3.3.11 – Cálculo da métrica bites *por pixel* (bpp) 62

Introdução

Motivação

Diz a sabedoria popular que uma imagem vale por mil palavras. Se esta é uma verdade insofismável, a sua era digital veio acrescentar o reverso da medalha. Se uma imagem vale por mil palavras, estas ocupam o espaço de muitos milhares de palavras.

Na emergente era digital, as imagens passam a ter o seu suporte favorito no formato digital, devido à facilidade de armazenamento, duplicação e transmissão. Longe vai o tempo, em que o armazenamento do álbum de família passava pela digitalização das fotografias. Hoje em dia, as fotografias são captadas, armazenadas e transmitidas em formato digital, sendo cada vez mais uma preciosidade as imagens impressas.

A par disso, desenvolveram-se aplicações e tecnologia que lidam exclusivamente com imagens digitais. A *Internet*, os satélites e as sondas espaciais permitem-nos observar imagens que são recolhidas, processadas e transmitidas em locais muito distantes dos olhos que as observam.

Devido à proliferação destes equipamentos e das aplicações que manipulam este tipo de dados, tornou-se imperioso a redução do volume de informação. O mais popular dos algoritmos de compressão de imagens é, sem dúvida, o algoritmo JPEG[1]. Grande parte da sua popularidade deve-se ao desenvolvimento da *Internet*, onde as limitações da largura de banda e o crescimento exponencial de utilizadores encontraram neste algoritmo um precioso aliado para a partilha de imagens digitais.

[1] Joint Photographic Experts Group

Fruto de investigações recentes, vários outros algoritmos têm vindo a ser propostos e desenvolvidos, como é o caso da técnica que propomos explorar neste trabalho.

São sobejamente conhecidas as imagens geradas por métodos fractais, devido à sua grande beleza e complexidade. A mesma tecnologia que dá origem a tão belas criações pode ser aplicada para comprimir imagens reais, e é sobre esta tecnologia que este trabalho se debruça.

Trabalhos anteriores

Nos últimos anos tem-se assistido a uma proliferação de publicações nesta área, sendo a esmagadora maioria dedicada a comprimir imagens monocromáticas. Algumas destes trabalhos fazem referências breves ao modo como se podem comprimir imagens coloridas. No entanto, não apresentam resultados práticos, e a complexidade intrínseca a este tipo de imagens tem sido tratada de uma forma ligeira e, por vezes incompleta.

Os trabalhos a que tivemos acesso permitiram-nos descortinar quatros métodos distintos de compressão de imagens coloridas com fractais, cujo desenvolvimento aparece patenteado ao longo desta dissertação.

Descrição do trabalho

O desígnio do presente trabalho consiste em investigar a área de compressão de imagens com fractais do tipo IFS, aplicada a imagens coloridas. Aos algoritmos que servem de base a este trabalho serão efectuadas melhorias de forma a incrementar a eficiência da compressão e a torná-los mais atractivos para aplicações reais.

Este trabalho tem dois objectivos principais: implementação dos métodos de compressão fractal adaptados às imagens coloridas; e avaliação do seu desempenho.

A técnica aqui explorada é complexa e com muitas variantes e parâmetros, de modo que, para conseguir o segundo objectivo, foi necessário que os métodos fossem aplicados às mesmas imagens e partilhassem os mesmos parâmetros e recursos.

O capítulo 1 descreve o funcionamento da tecnologia aplicada a imagens monocromáticas, desde o seu aparecimento até ao estado actual. Este capítulo serve de introdução à técnica aplicada às imagens monocromáticas. De forma a compreender melhor a complexidade acrescida da compressão de imagens coloridas, faz-se no capítulo 2 uma breve incursão sobre a temática da cor e a sua aplicabilidade às imagens digitais. Uma breve descrição do sistema visual humano, dos espaços para modelação de cor e a sua aplicação às imagens digitais são aqui fornecidas.

O capítulo 3 descreve os métodos implementados no nosso trabalho, e definem-se, supletivamente, as métricas para a fidelidade e compressão das imagens. Devido à grande quantidade de parâmetros, o capítulo 4 define os parâmetros a serem utilizados para a obtenção dos resultados práticos expressos no capítulo 5.

O capítulo 6 faz uma análise crítica dos resultados obtidos no capítulo anterior e sugere possíveis melhorias do nosso sistema a explorar futuramente.

Este trabalho é acompanhado de um registo informático em CD-ROM, onde se disponibilizam as aplicações desenvolvidas para este trabalho, bem como as imagens e os resultados dos testes descritos nesta obra, e outros que omitimos por imperativos de necessidade de síntese.

Capítulo 1: Compressão de imagens com fractais

Introdução

A geometria fractal foi dada a conhecer ao mundo aquando de publicação do livro *The Fractal Geometry of Nature* do matemático Benoit Mandelbrot, em 1977. Esta geometria entra em confronto directo com a geometria euclidiana, que vigorou durante mais de 20 séculos, ao afirmar que as criações da natureza, ao contrário das coisas criadas pela mão humana, não podem ser descritas com as construções geométricas existentes. As árvores, as montanhas, os flocos de neve, etc. possuem um detalhe impossível de descrever por segmentos de recta ou curvas. A geometria fractal introduz o conceito geométrico de *ad infinitum* e os informáticos encontraram neste conceito uma base matemática para produzir imagens com aparência realística.

John Hutchinson demonstrou em 1981 as propriedades de um sistema de equações iteradas (IFS - Iterated Function System) e a sua aplicação para a geração de imagens fractais.

Uma década mais tarde, Michael Barnsley em *Fractals Everywhere* [BAR88], desenvolveu um novo teorema, que ficou conhecido como o "teorema da colagem" que define as condições para que um sistema de equações possa descrever uma imagem. Este teorema abriu uma nova frente de investigação: a codificação de imagens através de fractais.

"Se um sistema de equações consegue produzir imagens com aparência realista, então dada uma imagem deverá ser possível encontrar um sistema de equações que a

possam descrever". Este problema é chamado de problema inverso. Barnslay em [BAR88] apresenta a primeira resolução para o problema inverso, mas conta com a intervenção humana para a sua solução final.

Solução do problema inverso - abordagem clássica

Uma forma de introduzir a notação e o funcionamento dos IFS, desenvolvidos por Hutchinson, é a utilização da metáfora da Máquina de Múltiplas Cópias Reduzidas (MMCR). Esta máquina é semelhante a uma copiadora normal com as seguintes modificações:

Tem múltiplas lentes que conseguem fazer várias cópias simultâneas do original, sendo o resultado final a composição de todas as cópias.

Cada lente tem a capacidade de reduzir, espelhar[2] e rodar o original.O resultado de cada lente pode ser trasladado para qualquer parte da cópia

A Copiadora opera em modo de *feedback*. O resultado da operação anterior é a imagem a copiar na iteração seguinte.

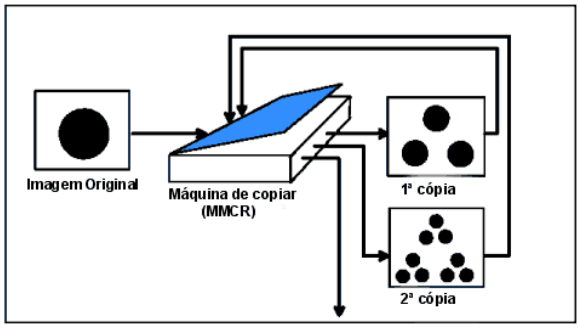


figura 1.1 - Máquina de Múltiplas Cópias Reduzidas [KOM95a].

A figura 1.2 mostra algumas das operações elementares que cada lente da MMCR consegue fazer. A acção final da lente pode conter qualquer combinação das operações elementares à qual se junta uma translação no espaço.

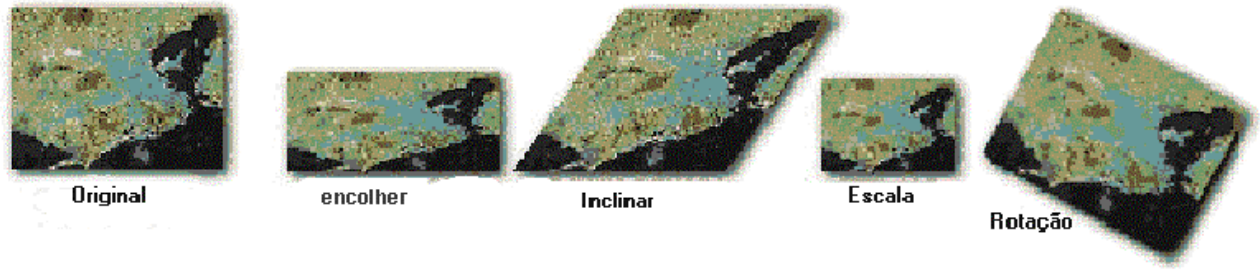


figura 1.2 – Imagem original e o resultado das operações permitidas pela MMCR [BRAN99] (adaptado).

[2] Espelhar significa inverter da imagem segundo o eixo do X ou do Y que funcionam como espelhos.

A MMCR apresentada na figura 1.1 tem no seu conjunto três lentes que produzem em cada iteração três imagens reduzidas do original e as deslocam para uma nova posição. A

figura 1.3 mostra as cinco primeiras iteração da MMCR.

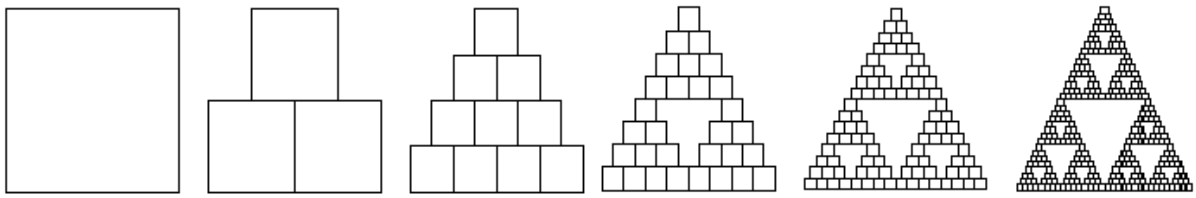


figura 1.3 – Imagem original e as 5 primeiras iterações da MMCR da figura 1.1 aplicadas a um quadrado.

Após mais algumas iterações a imagem produzida pela máquina mantêm-se invariável e chega-se à imagem apresentada na figura 1.4.

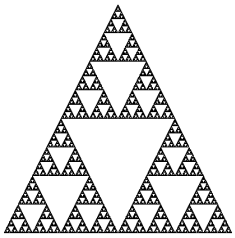


figura 1.4 - Triângulo de Serpinsky.

As propriedades contractivas das lentes da MMCR são uma propriedade fundamental para a obtenção de uma imagem final fixa. Esta imagem é frequentemente chamada **atractor** ou **ponto fixo**.

Cada lente pode ser representada pela equação $AX+T$, sendo *A* uma matriz quadrada que permite efectuar as operações da figura 1.2, *X* um ponto, na imagem original, e *T* a translação da cópia. A lente pode ser representada matematicamente pela equação 1.1.

$$w\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}$$

equação 1.1 - Transformação afim de um ponto no plano 2D

A matriz A contém os coeficientes necessários para a rotação e o factor de escala e pode ser rescrita pela equação 1.2, onde $(r1,\theta 1)$ são as coordenadas polares do ponto (a,c) e $(r2,(\theta 2+\Pi/2))$ são as coordenadas polares do ponto (b,d) apresentados na equação 1.1.

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} r1\cos\theta 1 & -r2\sin\theta 2 \\ r1\sin\theta 1 & r2\cos\theta 2 \end{pmatrix}$$

equação 1.2 - Matriz A (rotação, escala).

A figura 1.5 representa a aplicação de matriz A isolada a um quadrado definindo a escala (r_1 e r_2) e a rotação (θ_1 e θ_2) para cada um dos eixos. Quando algum dos factores de escala for negativo faz o efeito de espelho.

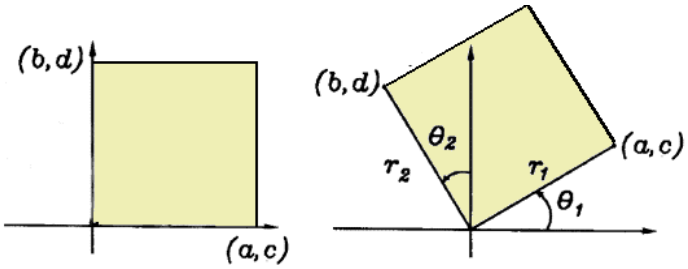


figura 1.5 – Aplicação da matriz A a um quadrado.

A figura 1.6 represente a aplicação de uma IFS completa (escala, rotação e translação).

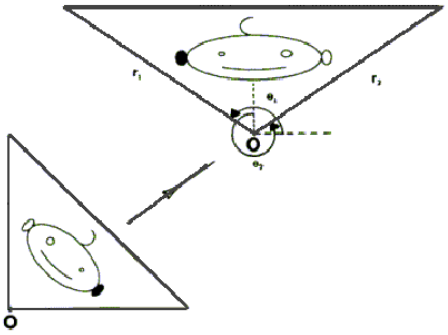


figura 1.6 – Aplicação de uma transformação afim IFS a uma figura [BAR88].

Num IFS, cada ponto da imagem original com coordenadas (x,y) é transformado num conjunto de novos pontos, com coordenadas (x_1,y_1), (x_2,y_2), . . . ($x_n$$y_n$) sendo *n* o número de transformações afim que compõem o IFS. Estes pontos são obtidos pela equação 1.3 .

$$\begin{pmatrix} x_1' \\ y_1' \end{pmatrix} = w_1 \begin{pmatrix} x \\ y \end{pmatrix}$$
$$\begin{pmatrix} x_2' \\ y_2' \end{pmatrix} = w_2 \begin{pmatrix} x \\ y \end{pmatrix}$$
$$\ldots$$
$$\begin{pmatrix} x_n' \\ y_n' \end{pmatrix} = w_n \begin{pmatrix} x \\ y \end{pmatrix}$$

equação 1.3 - Transformação pontual do IFS composto pelas transformações afim w_1, w_2, ... w_n.

$$\left[ab - cd\right] < 1.0$$

equação 1.4 – Critério de convergência de uma tranformação afim w.

A convergência do IFS é assegurada se, para cada função afim *wi,* a inequação 1.4 for respeitada [BAR88]. A figura 1.4 pode ser produzida através do seguinte conjunto de equações afim:

$$w_1 = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad w_2 = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} \quad w_3 = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 0.25 \\ 0.5 \end{bmatrix}$$

equação 1.5 - Conjunto de transformações afim , IFS , que geram o triângulo de Serpinsky.

Da equação 1.5 facilmente se compreende que cada função afim w_i (correspondente a uma lente da MMCR) reduz a imagem original a metade e não efectua qualquer tipo de rotação. A diferença entre as equações afim está na translação efectuada por cada uma. Como cada uma das funções afins é contractiva[3] é de esperar que a imagem convirja para um único ponto fixo independentemente da imagem original.

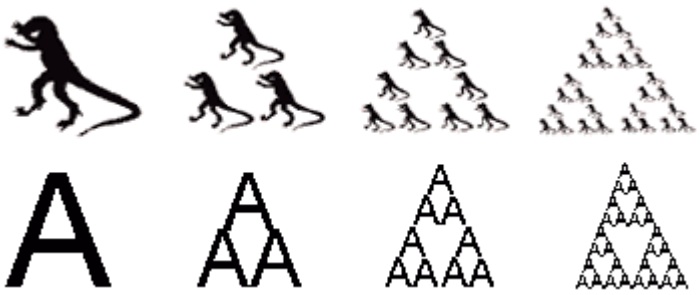


figura 1.7 – Primeiras três iterações obtidas através do IFS da equação 1.5 de duas imagens distintas.

Cada imagem é composta pela sobreposição das imagens produzidas por cada lente e se o critério de convergência for respeitado, obtemos o ponto fixo num número finito de passos. Cada imagem intermédia é obtida a partir da imagem obtida a partir do resultado da iteração anterior sendo a primeira obtida pela transformação da imagem original I_o .

$$\begin{aligned} I_1 &= w_1(I_o) \cup w_2(I_o) \cup \ldots \cup w_n(I_o) \\ I_2 &= w_1(I_1) \cup w_2(I_1) \cup \ldots \cup w_n(I_1) \\ &\ldots. \\ I_m &= w_1(I_{m-1}) \cup w_2(I_{m-1}) \cup \ldots \cup w_n(I_{m-1}) \end{aligned}$$

equação 1.6 – Imagens obtidas através de um IFS.

Como se pode verificar na equação 1.6, cada nova iteração dá origem a uma nova imagem que contém cada vez detalhes mais finos. Também se pode verificar na figura

3 Respeita o critério de convergência (equação 1.4)

1.7 que a imagem produzida em cada iteração é auto-similar com a anterior, uma vez que cada w_i faz uma cópia reduzida da imagem de entrada. O processo para a obtenção da imagem final I_∞ é descrito pela equação 1.7.

$$I_1 = w_1(I_o) \cup w_2(I_o) \cup \ldots \cup w_n(I_o) = \bigcup_{i=0}^{n} w_i(I_0) = W^1(I_o)$$

$$I2 = w_1(w_1(I_o)) \cup w_2(w_2(I_o)) \cup \ldots \cup w_n(w_n(I_o)) = \bigcup_{i=0}^{n} w_i(W^1(I_o)) = W^2(I_o)$$

$$I_\infty = \lim_{m \to \infty} W^m(I_0)$$

equação 1.7- Imagem final do IFS

Pressupostos para a compressão de imagens através de um IFS

A compressão de imagens por fractais assenta no princípio de que se um conjunto de funções pode gerar imagens, então através das imagens podemos chegar ao conjunto de equações.

O processo de compressão de imagens com fractais assenta nos seguintes pressupostos:

1. Muitas das imagens recolhidas na natureza possuem detalhes dentro de detalhes na sua composição, e, portanto, podem ser descritas pela geometria fractal;
2. Um IFS permite gerar imagens fractais que representam cenas com aspecto realístico;
3. Os coeficientes de cada transformação afim podem ser armazenados de forma compacta;
4. É possível a obtenção das transformações afins a partir de uma imagem qualquer.

Ponto 1

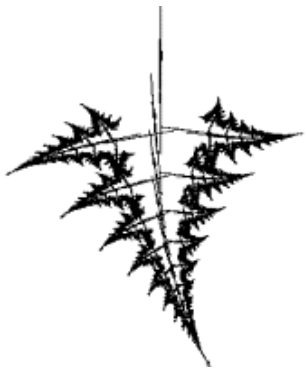
A verdade da primeira suposição está claramente explorada no livro *Fractals Everywhere* [Bar88] , e pode ser verificada quando olhamos para uma folha de árvore ou um floco de neve, ou para qualquer outro objecto criado na natureza. Apenas os itens criados pela mão humana seguem linhas rectas e esquadrias perfeitas. As imagens da natureza são muito mais complexas, com detalhes mais finos e mais perfeitos, que as formas geométricas convencionais apenas podem descrever aproximadamente e de forma incompleta.

Ponto 2

As imagens seguintes apresentam algumas formas fractais e os respectivos códigos IFS que permitem validar o ponto 2. Nelas podemos reconhecer as semelhanças com as imagens naturais exploradas no ponto 1.

A rotação é apresentada em graus e ambos os ângulos são iguais. Para a conversão para transformações afins aplica-se a equação 1.2.

Atractor Final | **IFS**



Função afim (w)	Escala X	Escala Y	Deslocamento X	Deslocamento Y	Rotação (graus)
w1	0.009	0.432	-0.015	0.5	0.0
w2	0.35	0.38	0.08	0.485	77.0
w3	0.335	0.305	0.476	0.41	-80.0
w4	0.665	0.815	0.155	0.162	-4.0

Função afim (w)	Escala X	Escala Y	Deslocamento X	Deslocamento Y	Rotação (graus)
w1	0.97	0.97	-0.076	0.119	-11.0
w2	0.185	0.185	0.383	-0.002	88.0

Função afim (w)	Escala X	Escala Y	Deslocamento X	Deslocamento Y	Rotação (graus)
w1	0.035	0.59	0.484	0.0	0.0
w2	0.5	0.5	0.25	0.0	0.0
w3	0.5	0.5	0.651	0.29	45.0
w4	0.5	0.5	0.0	0.642	-45.0

Função afim (w)	Escala X	Escala Y	Deslocamento X	Deslocamento Y	Rotação (graus)
w1	0.78	0.74	0.107	0.27	-2.0
w2	0.01	0.35	0.5	0.0	0.0
w3	0.28	0.35	0.413	0.314	-50.0
w4	0.28	0.28	0.423	0.089	52.0

figura 1.8 – Imagens obtidas através de IFS.

Muitas outras imagens poderiam ser aqui expostas. No entanto, as imagens da figura 1.8 validam, na perfeição, este ponto.

Ponto 3

Sem qualquer preocupação acerca de uma representação eficiente dos coeficientes, podemos fazer umas contas rápidas acerca do triângulo de Serpinsky apresentado na figura 1.4. Para a sua obtenção, são necessárias três transformações, cada uma com seis coeficientes. Se cada coeficiente for representado em dois bytes temos um tamanho de 36 bytes. Se a imagem produzida for de dimensão 512 por 512 pixels e se cada pixel for representado por um bit temos uma compressão de aproximadamente 1:910 o que é uma taxa de compressão excepcional.

Uma vez que o código para a geração do fractal não tem dimensões pode dar origem a imagens com qualquer dimensão. Como evidência deste facto podemos reparar que nas equações que geram as imagens que ilustram o ponto 3, os coeficientes da translação e do factor de escala estão expressos no intervalo [0..1] funcionando como uma fracção do tamanho original, logo, cada uma daquelas equações pode gerar uma imagem de qualquer tamanho.

As mesmas equações do IFS da figura 1.8 podem gerar imagens tão pequenas ou grandes quanto se queira, convém no entanto salientar que, uma imagem grande precisa de mais iterações para a obtenção do ponto fixo.

Ponto 4

Este ponto constitui o verdadeiro problema da compressão de imagens e consiste na reversão da engenharia que faz a geração de fractais. A resolução deste problema representa a descrição da imagem através dos coeficientes das funções afim que a compõem. A compressão de imagem é, no fundo, a resolução deste ponto.

Barnslay em [BAR88] apresenta alguns método para encontrar as funções afim que deram origem às imagens geradas por IFS, no entanto o nosso problema é um pouco mais complexo, pois as imagens a serem comprimidas não são geradas por funções, mas antes recolhidas do mundo real.

Abordagem clássica – Algoritmo de Barnslay

Esta abordagem baseia-se no teorema da colagem desenvolvido em [BAR88]. Para comprimir uma imagem, basta fazer o processo inverso da geração de imagens através de IFS. Para tal, consideramos a imagem a comprimir como o ponto fixo do IFS e vamos procurar as funções que lhes deram origem.

Para conseguir este objectivo, temos de procurar partes da imagem que sejam semelhantes com a imagem toda. O objectivo da compressão é encontrar um conjunto finito de regiões deste tipo que cubram na totalidade a imagem original. O algoritmo não é totalmente automático e conta com a ajuda de um utilizador para desempenhar algumas tarefas.

O algoritmo começa por calcular uma imagem reduzida (Ir_i) da imagem original (I), onde o utilizador tem a liberdade de efectuar as operações permitidas pelas transformações afim (rotação e escala) e uma região R_i (translação) que seja similar a Ir_i dando origem a uma transformação afim cujos coeficientes são calculados pelo computador e são armazenados pois constituem uma parte da imagem comprimida.

De seguida, o algoritmo apresenta ao utilizador a área da imagem ainda não coberta pelo conjunto das transformações encontradas, e o processo continua até a imagem obtida pela sobreposição de todas as imagens reduzidas transformadas pelas equações afim calculadas ser menor do que o erro admissível pelo utilizador.

A intervenção humana passa exactamente pela descoberta da rotação, escala e translação de cada imagem *Ir*, ficando a cargo do computador o cálculo dos coeficientes de cada IFS. Desta maneira, são calculadas as transformação *wi* que, quando iteradas dão origem a uma imagem que será próxima da imagem original.

Para avaliar qualidade da imagem obtida pela compressão, Barnslay utilizou a medida de Hausdorf [BAR88]. Quando esta distância for suficientemente pequena, os coeficientes das transformações wi são guardados e formam a imagem compactada. A seguir é apresentado um exemplo de compressão através deste método.

figura 1.9 – Imagem a comprimir.

A tabela 1.1 apresenta uma codificação possível da figura 1.9.

Transformação	Imagem da transformação	Transformação afim (Ax+T)	Imagem Coberta
w1		A: 0.6 0.0 / 0.0 0.6 — T: 0.18 / 0.12	

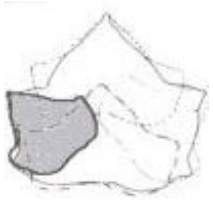
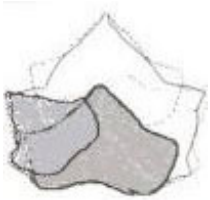
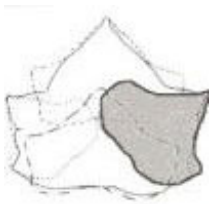
Transformação	Imagem da transformação	Transformação afim (Ax+T)	Imagem Coberta
w2		A T -0.4 0.3 0.27 0.3 0.4 0.09	
w3		A T 0.6 0.0 0.18 0.0 0.6 0.36	
w4		A T 0.4 0.3 0.27 -0.3 0.4 0.37	

tabela 1.1– Conjunto de transformações afim (IFS) que comprimem a figura 1.9.

O ficheiro da imagem comprimida teria o aspecto da figura 1.10.

0.6	0.0	0.0	0.6	0.18	0.12	-0.4	0.3	....	0.4	0.27	0.37

figura 1.10- Imagem comprimida.

Abordagem contemporânea – Algoritmo Jacquin

Jacquin foi o primeiro investigador a publicar um algoritmo totalmente automático para a compressão de imagens com IFS. A abordagem seguida por Jacquin segue uma filosofia significativamente diferente da anteriormente explorada. Ao contrário do algoritmo de Barnslay, que trabalha sempre com uma imagem reduzida da original, este algoritmo propõe que se divida a imagem em pequenas regiões (R_i), não sobrepostas, e que cubram a totalidade da imagem original. As funções afins que constituem a compressão são calculadas automaticamente, encontrando dentro da imagem original uma região maior (D_j) que seja a mais parecida possível com cada um dos blocos Ri. O facto dos blocos D_j serem maiores do que os blocos R_i garante que as transformações sejam contractivas.

figura 1.11 – Região autosimilar com um domínio na imagem Lena

De modo a aumentar a eficiência do algoritmo de compressão e descompressão, os blocos D_j têm o dobro do comprimento e da altura dos R_i e são admissíveis apenas as transformações da tabela 1.2. Como os blocos D_j têm o dobro do tamanho dos blocos Ri a sua comparação só é possível se estes forem reduzidos a metade. Como o factor de escala é significativo, o atractor é conseguido com poucas iterações.

figura 1.12 - Imagem que vai se submetida às transformações

Nº	Descrição da transformação	Matriz[4]		Imagem da Região R_i
1	Apenas Redução	1/2	0	
		0	1/2	
2	Redução e Reflexão no eixo do Y	-1/2	0	
		0	1/2	
3	Redução e Reflexão no eixo do X	1/2	0	
		0	-1/2	

4 As matrizes estão definidas para uma imagem com dimensões unitárias e não contêm a translação necessária para trazer a imagem para o primeiro quadrante.

Nº	Descrição da transformação	Matriz[4]		Imagem da Região R_i
4	Redução e Reflexão no eixo do Y=X	1/2	0	
		0	1/2	
5	Redução e Rotação de 90º	0	1/2	
		-1/2	0	
6	Redução e Rotação de 90º	0	1/2	
		-1/2	0	
7	Redução e Rotação de 180º	1/2	0	
		0	1/2	
8	Redução e Rotação de 270º	0	-1/2	
		1/2	0	

tabela 1.2– Tabela de transformações de Jaquin (rotação e escala).

A vantagem da utilização da tabela 1.2 para codificar a matriz A advém do facto de serem necessários apenas 3 bits.

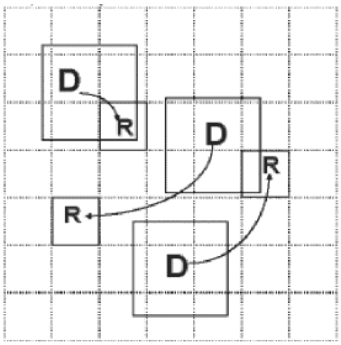


figura 1.13 - Representação gráfica do algoritmo de Jaquin

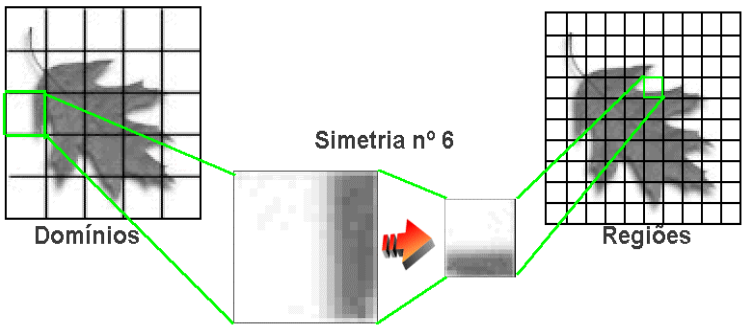


figura 1.14 - Codificação de uma região

A imagem comprimida é portanto um conjunto de coordenadas de domínios e das respectivas simetrias que codificam cada uma das regiões, tendo o ficheiro tantos conjuntos quanto os números de regiões Ri.

Extensão para a escala de cinza

Uma imagem em escala de cinza pode ser considerada uma superfície tridimensional onde a cada ponto de coordenadas (x, y) está associado o valor da intensidade do pixel. Uma imagem é um conjunto de elementos do tipo (x,y,f(x,y)) sendo f(x,y) a intensidade do pixel (x,y).

Para podermos tratar este tipo de imagens é necessário modificar a forma das funções afim de modo a poderem conter este novo valor. As transformações afim vão não só modificar as coordenadas dos pontos mas também modificar a sua intensidade, e podem ser reescritas da seguinte forma:

$$wi = w\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & s \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e \\ f \\ o \end{bmatrix}$$

equação 1.8 – Transformação afim com extensão à escala de cinza.

A forma como calculamos as coordenadas do novo ponto permanece inalterada. Sabendo qual a simetria do bloco, basta multiplicar os seus pixels pela matriz expressa na equação 1.8. Todavia, é necessário encontrar uma forma de calcular a nova intensidade. Neste novo modelo de imagens temos de encontrar uma nova métrica para medir a distância entre duas imagens em escala de cinza, uma vez que a métrica de Hausdorf funciona apenas para imagens binárias.

A métrica RMS[5] apresenta-se como uma boa candidata pois é simples e eficiente.

A distância entre a região R_i e o domínio D_j depois de transformado pela simetria S_k[6] pode ser expressa equação 1.9.

$$D_t = S_k \circ D_j$$

equação 1.9 – Obtenção do domínio transformado (Dt) através da simetria Sk

$$rms = \sqrt{\sum_{\forall x}\sum_{\forall y}(R_i(x,y) - D_t(x,y))^2}$$

equação 1.10– Distância entre a região R_i e o domínio D_j

Os valores ***s*** e ***o*** da equação 1.8 representam o factor de escala e o deslocamento que vai ser aplicado à intensidade de cor de cada pixel da região R_i. Fazendo $a_1,a_2,...,a_n$ os pixels do bloco de domínio D_t obtido pela equação 1.9, e $b_1,b_2,...,b_n$ os pixels do bloco da região R_i, necessitamos de procurar o factor de escala ***s*** e o deslocamento ***o*** que minimizam a distância entre os pixels do bloco do domínio e os pixels do bloco da região.

$$R = \sum_{i=0}^{n}((s * a_i^2 + o) - b_i)^2$$

equação 1.11 – Quadrado da diferença entre a região e o domínio

[5] Root Means Square

[6] Simetrias expressas na tabela 1.2

Derivando a equação 1.11 em relação a ***s*** e a ***o*** e calculando os zeros obtemos os valores para os quais a distância é mínima.

$$s = \frac{n * \sum_{i=0}^{n} a_i * b_i - \sum_{i=0}^{n} a_i * \sum_{i=0}^{n} b_i}{n * \sum_{i=0}^{n} a_i^2 - (\sum_{i=0}^{n} a_i)^2}$$

equação 1.12- Factor de escala para a intensidade do pixel

$$o = \frac{1}{n} * (\sum_{i=0}^{n} b_i - s * \sum_{i=0}^{n} a_i))$$

equação 1.13 - Deslocamento para a intensidade do pixel

Quando $n * \sum_{i=0}^{n} a_i^2 - (\sum_{i=0}^{n} a_i)^2 = 0$ então a escala é zero e o deslocamento é dado pela equação 1.14.

$$o = \frac{1}{n} * \sum_{i=0}^{n} b_i$$

equação 1.14 – Deslocamento quando a escala for zero.

A distância entre as duas imagens é dada pela métrica rms.

$$rms = \sqrt{\frac{R}{n}} = \sqrt{\frac{1}{n} \sum_{i=0}^{n} ((s * a_i + o) - b_i)^2}$$

equação 1.15 - Distância entre o domínio e a região (métrica rms)

Com a definição de uma métrica para poder comparar imagens e o cálculo dos coeficientes que possibilitam a transformação da intensidade dos pixéis, o algoritmo de compressão de imagens em tons de cinza funciona de modo idêntico ao das imagens binárias.

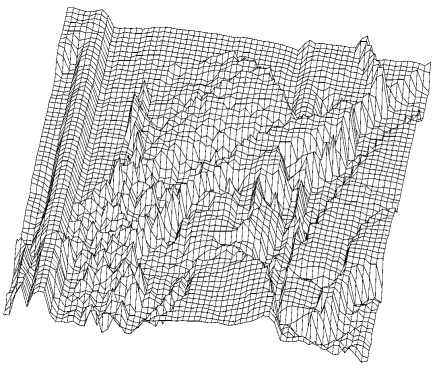


figura 1.15 – Representação tridimensional da imagem Lena [FIS95] .

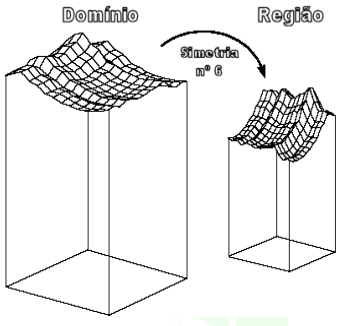


figura 1.16 – Codificação de um região tridimensional [GIG97] (adaptado)

Etapas da compressão com fractais

O algoritmo proposto por Jacquin constitui um grande avanço na tecnologia de compressão de imagens com IFS, e vários trabalhos foram desenvolvidos no sentido de melhorar o algoritmo proposto por ele.

Podemos dividir o problema da compressão de imagens com fractais em quatro sub problemas:

- Segmentação da imagem em regiões (Ri);
- Segmentação da imagem em domínios (Dj);
- Procura do bloco Dj e da transformação Sk para cada um dos Ri;
- Descodificação.

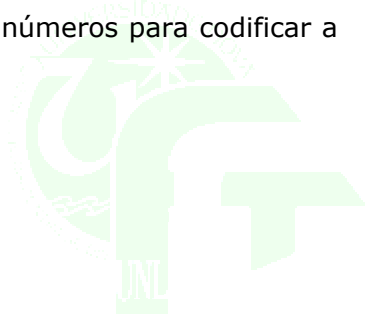
Os algoritmos actuais de compressão de imagens não diferem muito do apresentado por Jacquin, implementando novas técnicas para cada uma das fases da compressão.

Segmentação da imagem em regiões

A grande diferença entre a abordagem actual e a seguida por Barnslay é a utilização de transformações IFS aplicadas a regiões da imagem chamadas de PIFS[7], que ao contrário das anteriores(aplicáveis à imagem na totalidade), são aplicadas a uma determinada região. Para a divisão de imagem em regiões existem diversos métodos com vários níveis de eficiência e complexidade.

Partição Fixa

Este é o método mais simples de todos. A imagem é dividida em blocos de dimensão *nxm* usualmente 4x4 ou 8x8. Este método tem a grande vantagem de obtermos uma taxa de compressão fixa e a simplicidade do tratamento dos blocos, contudo, esta partição ignora por completo a imagem que estamos a tratar, e regiões com detalhe diferente são codificadas da mesma maneira, o que dá origem a erros de fidelidade, principalmente com partições maiores, ou uma baixa taxa de compressão com partições menores. Este tipo de partição tem ainda a vantagem de não ocupar espaço na imagem comprimida, uma vez que são necessários apenas dois números para codificar a partição.



7 Partioned Iterated Function System – Sistema de transformações afim que actuam apenas numa região da imagem.

Partição dinâmica

Com vista a resolver os problemas da partição fixa, foram desenvolvidos vários métodos para a adaptação da partição à imagem a tratar. Esta adaptação passa por construir uma árvore de blocos, onde cada bloco dá origem a vários blocos. A grande vantagem desta abordagem é o aumento da fidelidade da imagem comprimida, pois as regiões com mais detalhe são codificadas com blocos de menores dimensões, e, de modo inverso, os blocos onde a variância dos seus pixels é pequena são codificados com regiões maiores.

Existem duas abordagens possíveis para a construção da árvore: a abordagem *cega* e a abordagem *heurística*. Ambas as abordagens necessitam da definição *a priori* da quantidade de níveis que a árvore contém.

A abordagem *cega* começa por dividir a imagem em regiões com blocos do tamanho máximo. Quando uma região não encontra no domínio um bloco com uma semelhança admissível, divide-se o bloco em blocos mais pequenos e o processo continua iterativamente até a região atingir o tamanho mínimo ou até ser encontrado um bloco de domínio cuja semelhança com a região seja aceitável.

Diz-se que a abordagem é *cega* porque procura codificar os blocos com dimensões maiores independentemente da probabilidade dessa codificação ser feita.

Uma abordagem mais eficiente do algoritmo anterior é assumir a divisão *a priori*, através de uma função que nos indique a probabilidade de determinado bloco poder ser codificado ou não, em vez de procurar exaustivamente o domínio para depois decidir a divisão. Uma boa função heurística é a variância dos pixéis, pois blocos com uma baixa variância tendem a ser mais facilmente codificados com blocos maiores. A construção da árvore pode começar pelas folhas, agrupando blocos contíguos em blocos maiores aproveitando os valores dos blocos maiores para construir a heurística dos blocos maiores.

Os blocos podem assumir uma forma geométrica arbitrária. As partições a seguir apresentadas diferem apenas na forma dos blocos, podendo ser codificadas com qualquer uma das abordagens anteriores.

Quadtree

Os blocos tratados por esta partição são sempre quadrados, o que faz com que a divisão de um bloco dê origem a quatro novos blocos no nível imediatamente inferior. A grande vantagem desta codificação é o facto de bastar um bit para codificar se um bloco está ou não dividido. As desvantagens deste método resultam do facto dos blocos serem sempre quadrados independentemente da geometria da imagem a codificar e as imagens a codificar devem ser múltiplas do tamanho dos blocos menores.

Horizontal-Vertical(H-V)

A partição H-V é mais geral do que a anterior uma vez que trata imagens com quaisquer dimensões. Cada bloco pode ser dividido segundo um segmento de recta horizontal ou vertical, dando origem a dois novos blocos. A posição de cada um destes segmentos pode ser ajustada às características da imagem, incrementando deste modo a fidelidade da compressão, mas de modo inverso a representação desta partição necessita de mais bites e por consequência na redução da taxa de compressão.

Regiões irregulares

Nada obriga que as regiões sejam quadriláteros. Este tipo de partições amplifica as vantagens e desvantagens da partição H-V: aumenta a fidelidade e diminui a taxa de compressão.

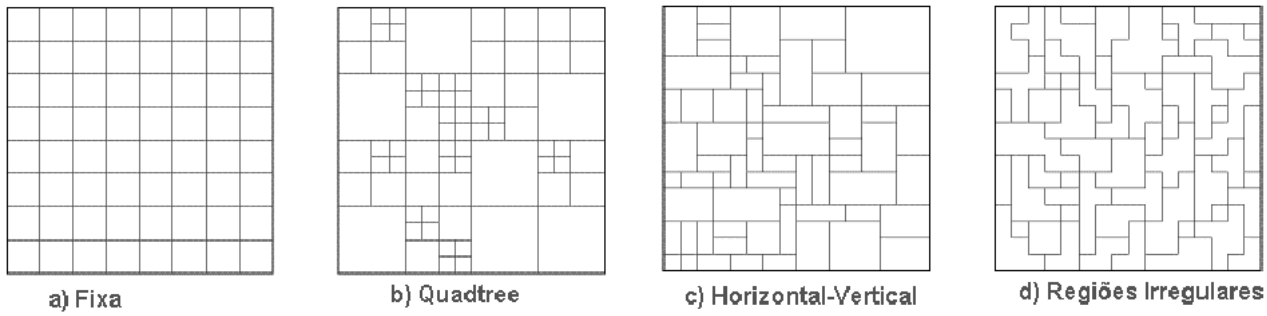


figura 1.17 –Partições fixa e dinâmica (Quadtree, Horizontal-Vertical e Regiões Irregulares).

Outros Polígonos

A utilização de regiões com ângulos rectos tem a vantagem da codificação ser compacta e da separação dos pixels para cada região ser imediata, contudo, existem outras abordagens que utilizam outro tipo de polígonos para descrever a partição. A abordagem através de triângulos é a mais conhecida e explorada.

O algoritmo começa por dividir a imagem em dois triângulos através de uma das diagonais da imagem. A divisão de cada um dos triângulos resultantes pode ser feita através da divisão de um, dois ou três lados do triângulo dando origem a dois, três ou quatro novos triângulos.

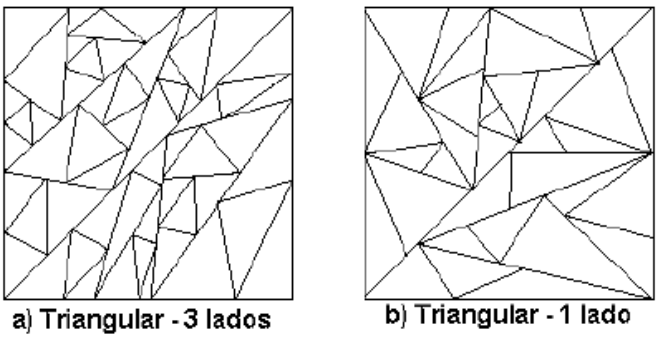


figura 1.18 – Partições triangulares.

Uma alternativa à divisão triangular é a abordagem seguida por Delaunay [DAV95]. A partição começa com um conjunto de pontos que se vão adaptar aos contornos da imagem, formando entre si polígonos. A este conjunto de pontos vão sendo adicionados novos pontos sempre que a variância numa região o justifique.

Uma partição semelhante consiste em fazer uma partição fixa e sempre que for necessário divide-se uma região através de um segmento de recta[8] dividindo-a assim em duas.

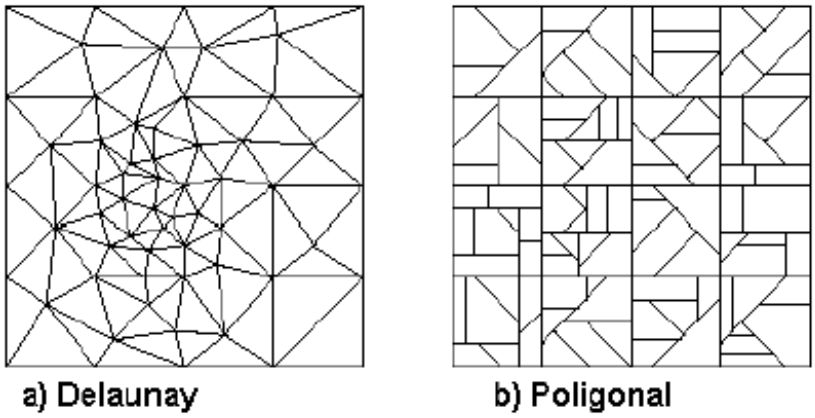


figura 1.19 – Partições poligonares

A seguir são apresentadas quatro partições distintias aplicadas à imagem Lena.

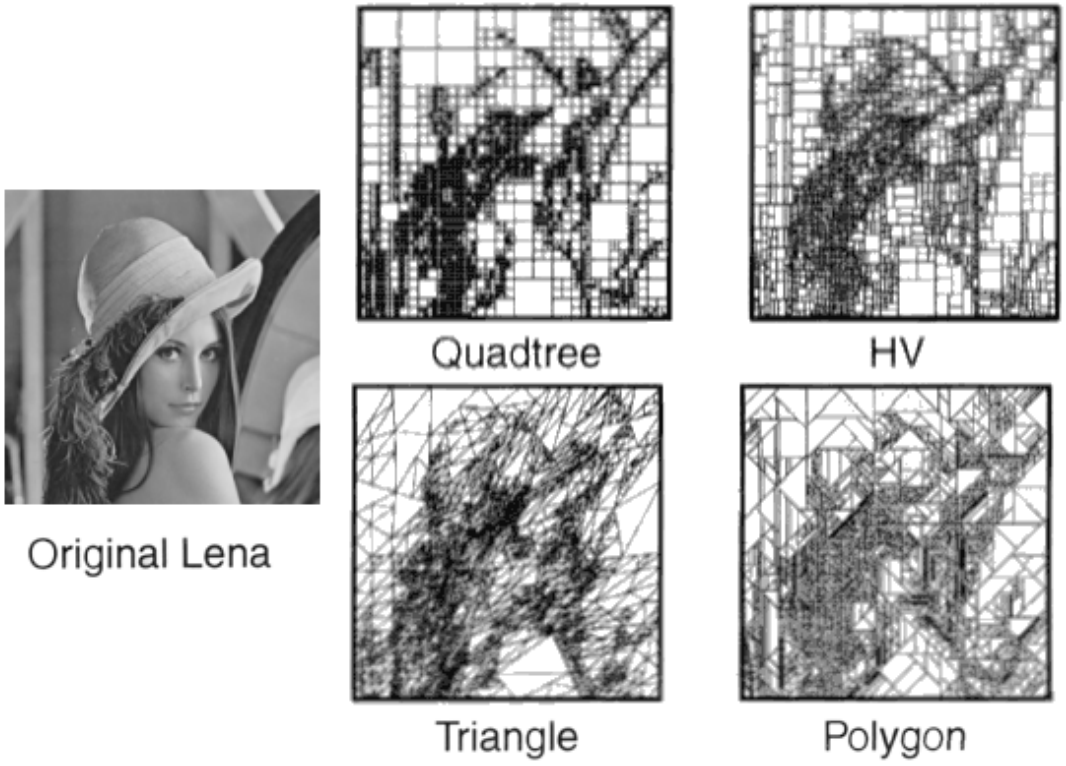


figura 1.20 – Partição da imagem Lena com quatro métodos distintos [www 1] (adaptado).

8 este segmento de recta pode ter qualquer inclinação e define-se entre dois lados da região.

Segmentação da imagem em domínios

A segmentação da imagem em domínios segue normalmente o mesmo algoritmo utilizado para a divisão da imagem em regiões. A utilização do mesmo algoritmo prende-se com o facto de ser mais fácil transformar uma região noutra com a mesma forma geométrica.

Para garantir o critério de convergência, utilizam-se normalmente domínios com o dobro do tamanho das regiões. Esta possibilidade, garante um factor de convergência eficiente, atingindo-se o atractor final com cerca de dez iterações.

Ao contrário das regiões, os domínios podem ser sobrepostos, e um mesmo domínio pode ser transformado para codificar várias regiões.

Domínios para a partição quadtree

Em *Fractal Image Compression* [FIS95], Yuval Fisher definiu três tipos de domínios para a partição quadtree, a que chamou D1, D2 e D3. Todos os domínios começam em pontos distanciados entre si de um comprimento fixo a que chamou grelha.

D1 – Os domínios em cada nível não são sobrepostos. Como é evidente existem mais blocos pequenos que blocos grandes devido à não sobreposição dos mesmos.

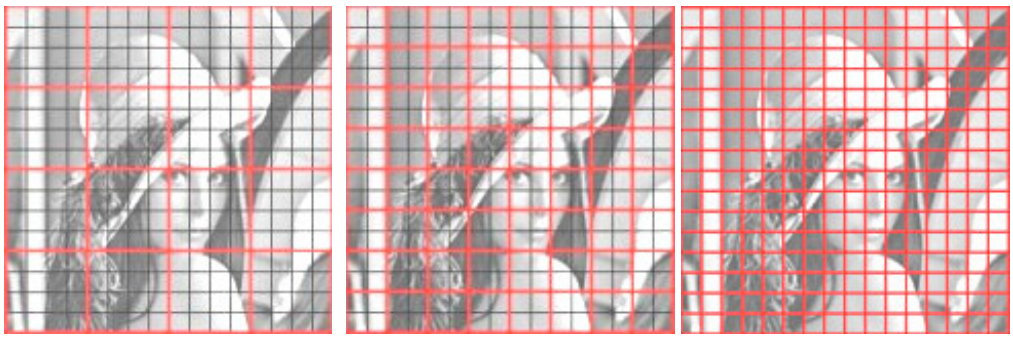


figura 1.21 – Quadtree de 3 níveis com domínios do tipo D1 para uma imagem 256x256 com grelha de tamanho 16.

D2 – Este tipo de partição é exactamente o inverso da anterior. Os blocos pequenos estão nas coordenadas dos blocos grandes da partição anterior, e os blocos grandes estão onde estavam os pequenos. As camadas intermédias são obtidas pelo mesmo processo. Este tipo de partição origina mais blocos grandes que pequenos.

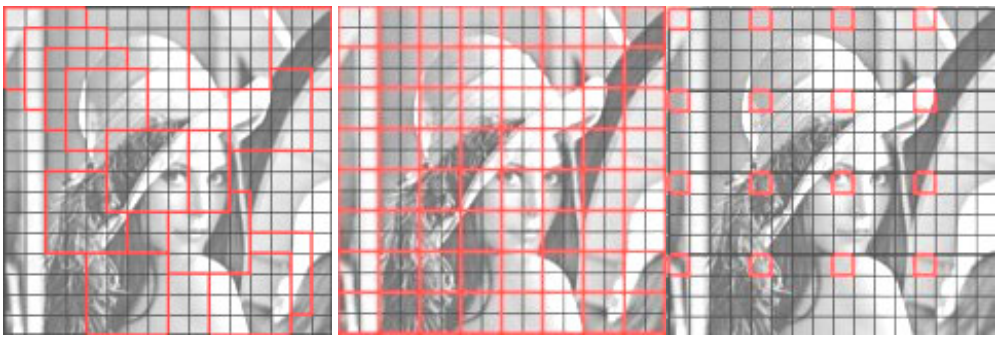


figura 1.22 - Quadtree de 3 níveis com domínios do tipo D2 para uma imagem 256x256 com grelha de tamanho 16.

D3 - Neste tipo de partição todos os domínios começam na grelha com espaçamento predefinido independentemente do seu tamanho. Este tipo de partição origina sensivelmente o mesmo número de blocos grandes e pequenos.

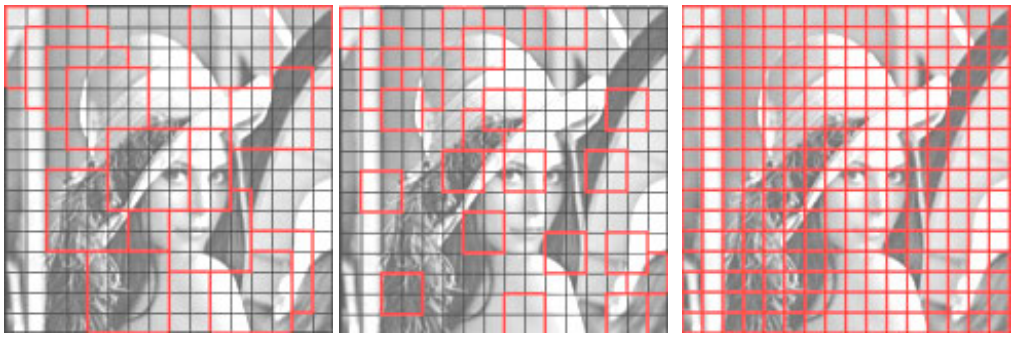


figura 1.23 - Quadtree de 3 níveis com domínios do tipo D3 para uma imagem 256x256 com grelha de tamanho 16.

Estratégias de procura

O maior esforço computacional para a compressão de imagens por fractais é utilizado na pesquisa do bloco do domínio que é mais próximo da região que pretendemos codificar.

Uma forma de acelerar este processo de procura, é fazer uma classificação dos domínios e das regiões de modo a comparar apenas aquelas que pertencem à mesma classe, ou a uma classe próxima.

Jacquin classificou os blocos em planos, sombreados e texturados[9]. Yuval Fisher em [FIS95] utilizou uma classificação mais elaborada, dividindo os blocos em classes de acordo com a sua média e variância. No método deste último, cada bloco é dividido em quatro regiões iguais, não sobrepostas sobre as quais são calculadas a média (A_i). De acordo com as médias A_i são classificadas em três classes principais sendo:

- Classe 1 – $A1 \geq A2 \geq A3 \geq A4$

9 flat, edge e textured

- Classe 2– $A1 \geq A2 \geq A4 \geq A3$
- Classe 3– $A1 \geq A4 \geq A2 \geq A3$

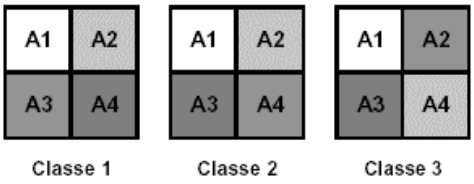


figura 1.24 – Classes principais [FIS95].

Cada uma destas classes dá origem a 24 subclasses ordenadas pelo desvio padrão de cada uma das regiões. Desta forma, podemos classificar os blocos com 3 classes principais e 72 subclasses. Se a escala ***s*** for negativa a ordem de cada classe é reorganizada. Todavia, cada bloco do domínio pode ser classificado com uma classe para a escala ***s*** positiva e outra para ***s*** negativa.

Uma classificação com muitas classes acelera o processo de compressão, mas limita o número de blocos que possam codificar uma região e, consequentemente, diminui as hipóteses de compressão com qualidade.

Descodificação

A descodificação da imagem consiste iterar o sistema de equações W sobre uma imagem inicial até esta atingir o ponto fixo (imagem descodificada).

Cada bloco da partição das regiões é substituído pelo respectivo bloco de domínio depois de modificado pela transformação afim. Tal como atrás foi referido, esta transformação inclui uma contracção, uma operação de simetria e para cada um dos pixels a modificação da sua intensidade através do factor de escala **s** e do deslocamento **o**.

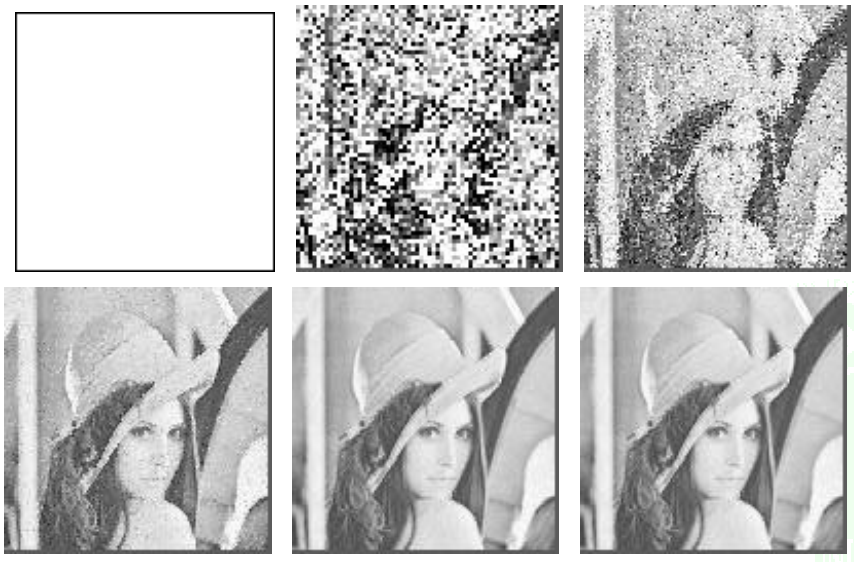


figura 1.25 – 5 Iterações da descodificação de imagem lena.

À imagem obtida pelo processo de descodificação poderão ser aplicados algoritmos de pós-processamento no intuito de melhorar a qualidade da imagem final.

Considerações finais

Quando comparada com outras técnicas de compressão de imagens a compressão com fractais é relativamente imatura, encontrando-se ainda algumas das fases em processo de investigação e desenvolvimento.

Este tipo de compressão é muito lento, embora a descompressão seja rápida. No entanto nos últimos anos tem-se assistido a um grande desenvolvimento a nível do hardware capaz de suplantar a lentidão da compressão, par disso, tem-se assistido nos últimos anos a um crescimento acentuado do número de publicações nesta área tornando esta técnica mais madura e com vantagens práticas como mais à frente demonstramos.

Capítulo 2: A Cor

O sistema visual humano

A cor existe por causa de três entidades: a luz, o objecto visualizado e o observador. Fisicamente, a luz pode ser vista como um feixe de ondas electromagnéticas, composto por diferentes comprimentos de onda, que, quando embate num objecto, reflecte alguns destes últimos, que são captados pelo observador. A sensação de cor é a interpretação que o nosso sistema visual faz das frequências reflectidas pelo objecto e captadas pelo olho.

Sir Isaac Newton, fazendo passar um feixe de luz branca por um prisma de vidro, conseguiu separar os diferentes comprimentos de onda (figura 2.1), obtendo, assim, o espectro visível da cor para os seres humanos.

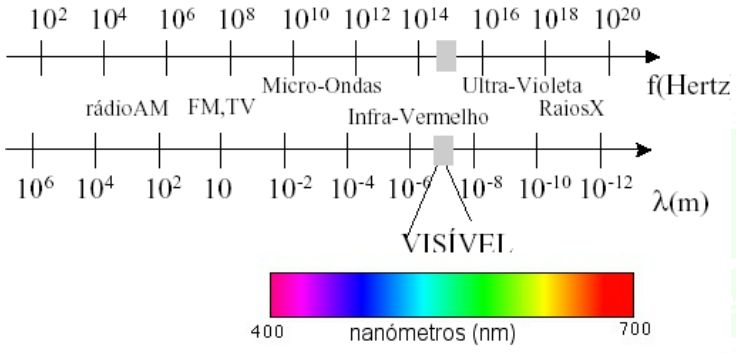


figura 2.1 - Comprimentos de onda da luz e o espectro visível [SCR99]

No nosso olho existem dois tipos distintos de sensores situados na retina: os cones e os bastonetes. Os bastonetes são sensíveis a todos os comprimentos de onda do espectro visível, dando uma visão perfeita e monocromática da forma (visão nocturna); no entanto, não são capazes de distinguir a cor. Os cones são menos sensíveis à intensidade da luz (visão directa), mas permitem a distinção de cores porque existem três tipos de cones distintos que respondem de forma distinta aos diferentes comprimentos de onda.

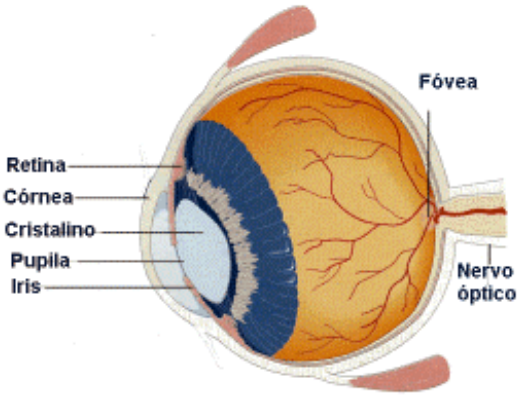


figura 2.2 - Sitema visual humano [ARE95] (adaptado)

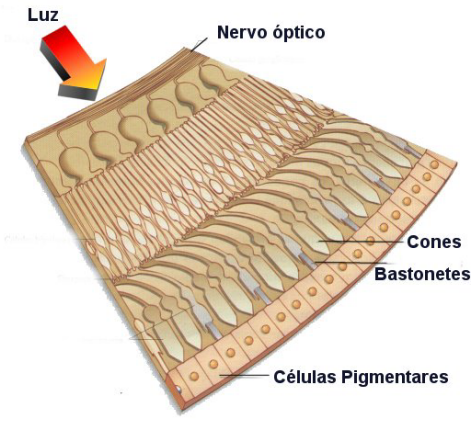


figura 2.3 - Ampliação da Retina [ARE95] (adaptado).

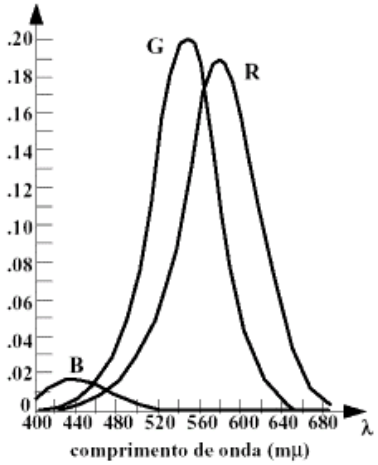


figura 2.4 – Fracção da luz absorvida por cada tipo de cone para as frequências do espectro visível [APO98].

A resposta de cada um dos cones às frequências do espectro visível está expressa na figura 2.4, que caracteriza o processo de discriminação de cor do olho chamado "tricromacidade".

O nosso sistema visual funciona de forma aditiva, não conseguindo distinguir os estímulos de cada tipo de cone isoladamente (ao contrário do nosso ouvido, que consegue distinguir diversos tipos de ondas em simultâneo, por exemplo, um piano e uma guitarra), mas apenas o seu resultado final, obtido pela composição dos três tipos.

Imagens digitais

Quando a luz incide num determinado objecto, alguns comprimentos de onda são reflectidos de acordo com o ângulo de incidência a geometria do objecto e a composição do mesmo. A luz reflectida é convertida em impulsos eléctricos pelo nosso sistema visual, transmitidos ao cérebro, que os interpreta como uma imagem.

Um processo semelhante ocorre com as máquinas fotográficas convencionais, que, com os seus sensores, captam a distribuição espacial da cor dando uma imagem a duas dimensões do mundo que está a ser fotografado.

São estes tipos de imagens que nos propomos tratar. Todavia, tanto a cor como a distribuição espacial da mesma são grandezas físicas representáveis por números reais, e o computador apenas trata de grandezas discretas, números inteiros e números reais com precisão finita.

Devido a este facto, a imagem captada do mundo real tem de passar por um processo de discretização e quantificação que a transforme noutra imagem, apesar de mais pobre, passível de ser tratada computacionalmente.

Discretização e quantificação

A discretização da imagem é vulgarmente chamada de digitalização e consiste em converter as dimensões e a cor da imagem em grandezas representáveis por números inteiros.

A discretização[10] é a divisão da imagem real, sem dimensões, em pequenos elementos, normalmente quadrados (pixels[11]), dando origem a duas dimensões: a horizontal e a vertical.

O processo de discretização dá origem a uma matriz bidimensional que vai ser preenchida pela cor de cada pixel. A este processo chama-se quantificação. Geralmente utilizam-se 8 bites para codificar cada uma das suas componentes, dando origem a 256 níveis de intensidade para cada componente. A combinação dos três componentes dá origem a imagens com um pouco mais de 16 milhões de cores.

Resultados experimentais demonstram que o olho humano consegue distinguir cerca de 400 mil cores, de modo que esta quantificação não introduz erros perceptíveis na imagem. A figura 2.5 e a figura 2.6 mostram o processo de discretização e quantificação de uma imagem.

[10] Este processo também é conhecido por amostragem.

11 Picture Element

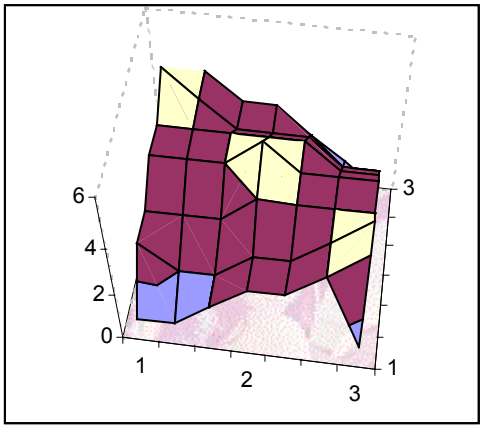


figura 2.5 - Imagem analógica.

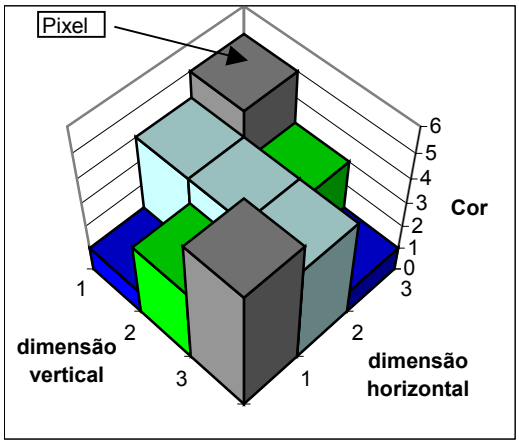


figura 2.6 - Imagem digital.

Modelação da cor

As imagens digitais, obtidas no mundo real, para terem interesse prático, precisam de voltar a ser reproduzidas ou visualizadas, extraindo a cor de cada um dos seus pixels.

Existem dois processos para a obtenção da cor: o processo aditivo e o processo subtractivo. No processo aditivo, os raios de luz combinam-se de modo a formar um novo raio. A cor pode ser obtida artificialmente através de dispositivos electrónicos com a conjugação das três frequências que fazem a tricromacidade. No processo subtractivo utilizam-se filtros para suprimir algumas frequências ao raio original. Estes filtros podem ser pigmentos, tintas ou dispositivos electrónicos.

O objectivo de um modelo é a representação das cores de uma forma normalizada e tem como objectivo servir de suporte para a quantificação da cor.

Os sistemas de cor a seguir especificados fazem parte do padrão definido pela CIE[12] e são apenas alguns dos mais utilizados, existindo diversas variantes para cada um deles.

Sistema XYZ

O processo aditivo não consegue formar todas as cores possíveis do espectro visível, pois, para ajustar determinadas cores, é necessário subtrair algumas componentes. A utilização de componentes negativas traz algumas desvantagens não só

12 Commission International de L'Éclairage

para a construção de hardware mas também para a representação eficiente da cor. A forma encontrada pelo CIE para contornar as dificuldades introduzidas pelos coeficientes negativos foi estender a escala dos componentes RGB para fora do espectro visível, dando origem a um sistema que possui cores que não são visíveis para o olho humano.

A figura 2.7 representa a pirâmide XYZ, que contém todas as cores e a parábola do espectro visível[13]. A figura 2.9 apresenta todas as cores do espectro visível, onde os contornos definem a área perceptível pelo olho humano.

A figura 2.8 apresenta um gráfico com a contribuição de cada uma das componentes RGB para a construção das cores do sistema XYZ. Estes dados são experimentais e foram calculados em 1931 (linha contínua) e posteriormente rectificados em 1964 (linha a tracejado) [SCR99].

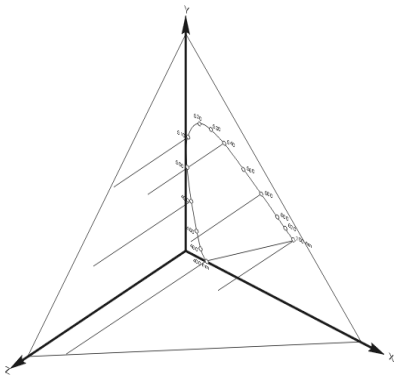


figura 2.7 – Pirâmide do sistema XYZ e o espectro visível [www 2].

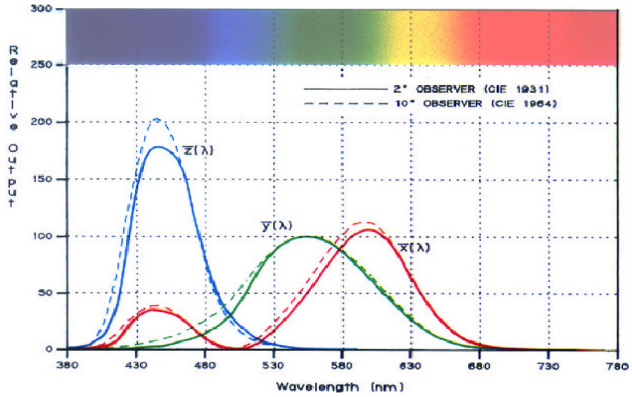


figura 2.8 - Reconstrução de cor do sistema RGB no XYZ - dados experimentais de 1931 e 1964 [SCR99].

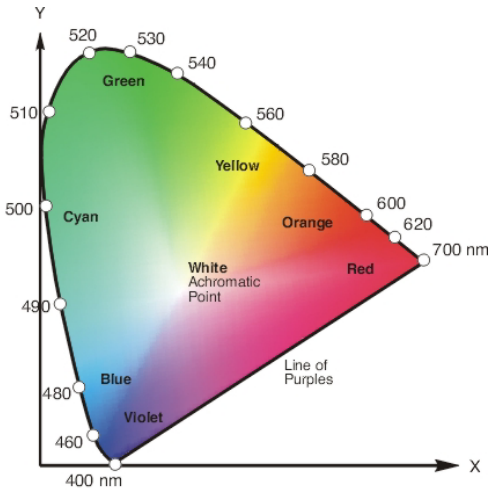


figura 2.9 – Cores visíveis Sistema XYZ (projecção) [www 2].

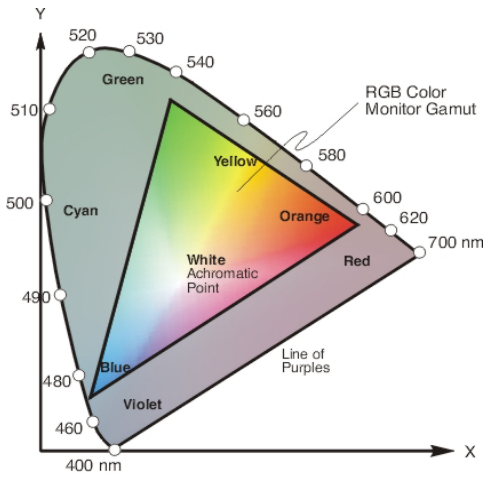


figura 2.10 – Cores disponíveis num monitor RGB típico [www 2].

[13] Projecção da coordenada Z no plano XY

O hardware de aquisição e visualização de imagem opera dentro de determinada região, também chamada *gamut*, dentro do espectro visível. A figura 2.10 mostra as cores disponíveis num monitor de raios catódicos RGB típico.

Sistema RGB[14]

A cor que nos interessa representar é destinada ao tratamento de imagens que são visualizadas por humanos. Desta maneira, a melhor forma de o fazer é tirar partido da tricromacidade dos nosso olhos e representar as cores como uma combinação linear das intensidades perceptíveis por cada um dos tipos de cones que possuímos.

Este modelo está baseado em coordenadas cartesianas tridimensionais, representando cada um dos eixos a intensidade da componente vermelha, verde e azul.

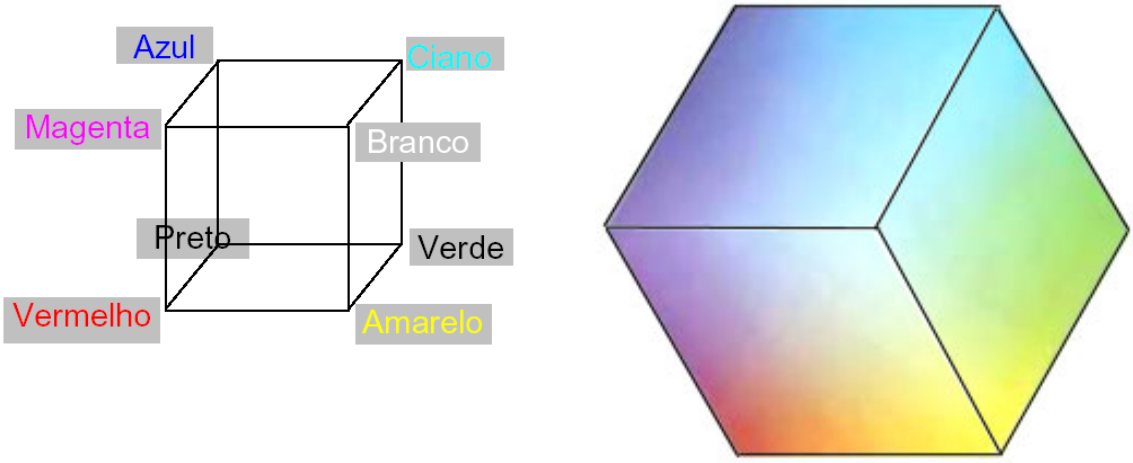


figura 2.11 – Cubo RGB [SCR99].

Este é um modelo aditivo, estando o preto na origem dos eixos e o branco, no vértice oposto do cubo.

Este sistema não possui todas as cores do espectro visível mas possui muitas mais do que aquelas que um utilizador tradicional consegue distinguir.

Este tipo de sistema utiliza-se, essencialmente, em hardware de captura e visualização de imagens, tal como monitores, câmaras fotográficas, scanners, etc.

14 RGB – Red Green Blue

Sistema CMY[15] e CMYK[16]

Como se pode verificar no cubo RGB, as cores ciano, magenta e amarelo são as cores complementares das cores RGB.

Este sistema de cor está intimamente ligado ao processo de formação de cores através do processo de subtracção. O sistema CMY pode ser visto como o cubo RGB invertido, sendo a origem dos eixos o branco.

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

equação 2.1 - Conversão do espaço RGB para CMY

A conversão do sistema RGB em CMY é dada pela equação 2.1. Esta equação.pressupõe que os coeficientes das componentes então normalizadas entre 0 e 1.

Este modelo é utilizado em impressoras onde ao papel branco (que reflecte todas as cores) são adicionados pigmentos que impedem a reflexão de determinadas cores. A junção dos três pigmentos na sua intensidade máxima fornece a cor preta, o que torna este processo muito dispendioso, por isso a maioria das impressoras utiliza o sistema CMYK sendo o K a cor preta, que é mais barata, evitando assim o desperdício dos outros pigmentos.

$$\begin{bmatrix} C \\ M \\ Y \\ K \end{bmatrix} = \begin{bmatrix} (C-K)/(1-K) \\ (M-K)/(1-K) \\ (Y-K)/(1-K) \\ mínimo(C,M,Y) \end{bmatrix}$$

equação 2.2 - Conversão de CMY para CMYK

O cálculo da componente K é feito através das componentes CMY e é normalmente obtido por cálculos mais complexos do que os definidos na equação 2.2 quando se pretende obter uma a impressão de alta qualidade.

Sistemas de Vídeo (YUV, YCrCb[17])

Estes sistemas baseiam-se na decomposição do sistema RGB em sistemas que utilizam a luminância e a crominância e estão ligados à emissão de imagens de televisão.

15 CMY – Cian, Magenta, Yellow.

16 CMYK – Cian, Magenta, Yellow, Black.

17 YCrCb – Luminância(Y), crominância vermelha (Cr) e crominância azul (Cb).

A luminância pode ser vista como a quantidade de luz que é emitida e a crominância, a cor que essa luz contém.

$$Y = \begin{bmatrix} 0.299 & 0.587 & 0.114 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

equação 2.3 - Cálculo da luminância através das componentes RGB.

As primeiras emissões de televisão foram monocromáticas e utilizavam o canal para a transmissão da luminância. Como se pode verificar na figura 2.4, a contribuição de cada uma das componentes não é uniforme, pois a componente verde e vermelha participam num espectro de cores mais alargado do que a cor azul. Devido a este facto, a conversão das três componentes RGB numa só componente (Y) não deve ter participação uniforme de cada uma das componentes.

Quando se introduziram as emissões de televisão coloridas houve a necessidade de introduzir a cor (crominância) e manter a compatibilidade com o sistema anterior. As componentes da crominância são obtidas subtraindo a luminância (Y) das componentes RGB e eliminando-se a componente G pois, como atesta a equação 2.3 é a responsável pela maior parte da luminância. Assim as componentes ficam Y, R-Y, e B-Y. Este sistema é chamado de vídeo composto porque as componentes são combinadas no mesmo canal.

A transformação do sistema RGB para YUV e pode ser representado pela equação 2.4

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

equação 2.4 - Transformação de RGB para YUV.

Para obter a transformação inversa basta inverter a matriz.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 1.000 & -0.714 & -0.343 \\ 1.000 & -0.000 & 1.772 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix}$$

equação 2.5 - Transformação de YUV para RGB.

Os sistemas de vídeo composto mais utilizados são o YUV para vídeo analógico e YCrCb para vídeo digital.

A conversão de e para os espaços YCrCb é obtida respectivamente, pela equação 2.6 e pela equação 2.7.

$$\begin{bmatrix} Y \\ Cr \\ Cb \end{bmatrix} = \begin{bmatrix} 0.255 & 0.502 & 0.097 \\ -0.147 & -0.289 & 0.437 \\ 0.437 & -0.366 & -0.071 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

equação 2.6 - Transformação de RGB para YCrCb.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.168 & 0.000 & 1.602 \\ 1.168 & -0.393 & -0.816 \\ 1.168 & 2.025 & 0.000 \end{bmatrix} \begin{bmatrix} Y \\ Cr \\ Cb \end{bmatrix}$$

equação 2.7 - Transformação de YCrCb para RGB.

Sistema HSV[18]

Seleccionar uma cor no cubo RGB pode ser uma tarefa árdua. Por isso, existe um sistema de coordenadas não lineares que se baseiam noutras características da cor que são a luminância, o tom e saturação. Este sistema é vulgarmente utilizado por programas de desenho e manipulação de imagens digitais.

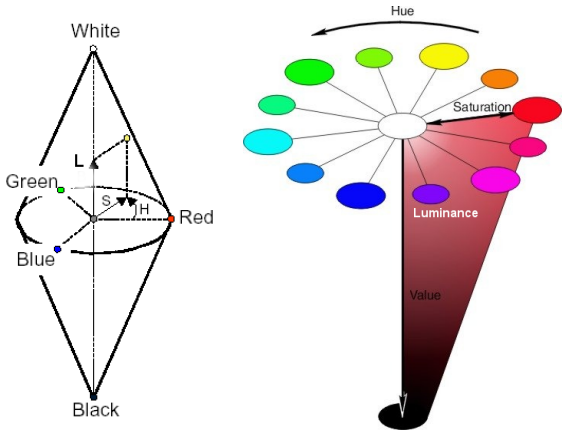


figura 2.12 – Cores no Sistema HSL (tom, saturação, luminância).

O cubo é transformado num duplo cone onde o eixo principal contém os valores da luminância. O plano perpendicular a este eixo fornece o valor da saturação. Esse plano fornece um círculo com todas as cores possíveis para cada nível de intensidade. Nesse nível, o tom é definido como o ângulo necessário para atingir a cor pretendida.

As fórmulas para a conversão de RGB para HSL e de HSL para RGB não são transformações lineares, e as suas fórmulas são um pouco mais complexas do que as conversões anteriores. Para mais detalhes acerca desta conversão, consultar a referência bibliográfica [GWO96].

18 HSL – Hue, Saturation, Luminance.

Sistemas enumerados

Uma alternativa para representar a cor é ter um conjunto enumerado e limitado de amostras de cores, onde cada cor é acessível através de um índice.

A figura 2.13 representa o sistema de Munsell. Existem muitos outros sistemas enumerados que variam no número de cores disponíveis e na correlação entre as cores.

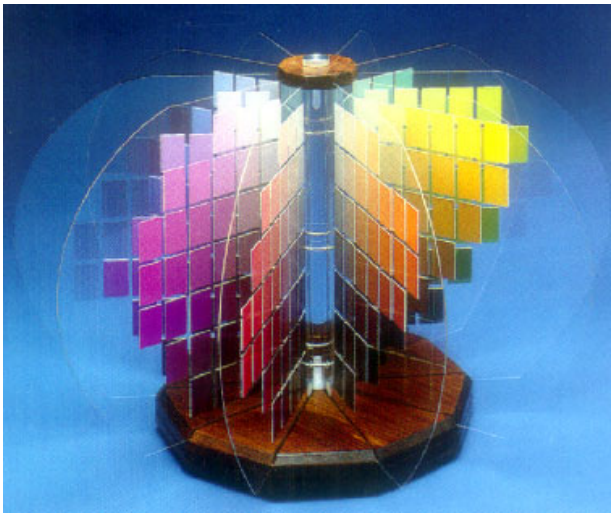
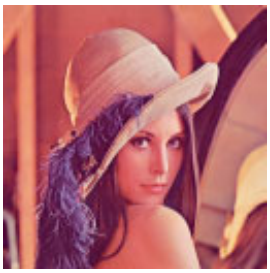


figura 2.13 - Sistema de Munsell (1905).

Representação de imagens nos diferentes espaços de cor

Normalmente, os dispositivos de aquisição de imagens digitais funcionam com o sistema de cor RGB. Contudo, para a transmissão, armazenamento ou visualização é comum proceder-se à mudança do espaço de cor. Os itens seguintes mostram a conversão da figura 2.14 de dimensão 128 x 128 pixels com 24 bites de cor, 8 para cada componente, do sistema RGB para cada um dos espaços de cor atrás descritos e vice-versa.

RGB	R	G	B
			

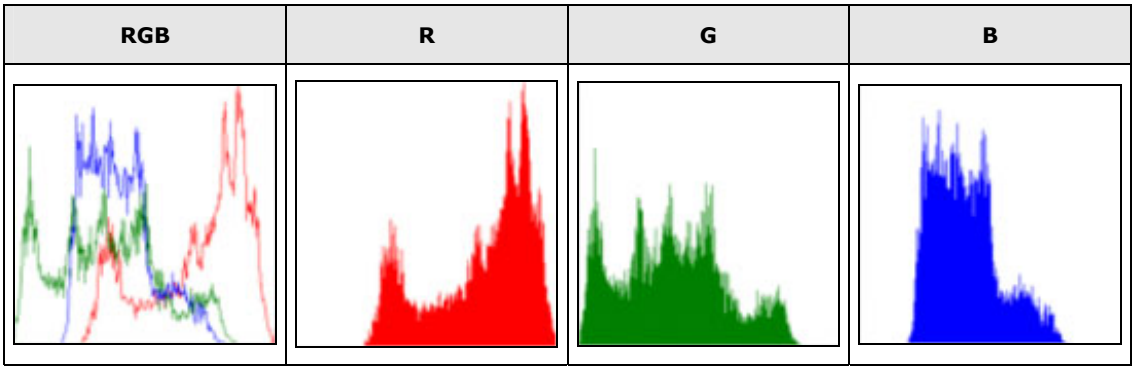


figura 2.14 – Imagem da Lena com 24 bites de cor no sistema RGB. Componentes e respectivos histogramas.

As componentes neste espaço estão altamente correlacionadas. Para imagens naturais, a correlação entre os níveis é de 0.78 entre o R e o B, 0.89 entre o R e o G e 0.94 entre o B e o G [HMS94].

Representação em CMY e CMYK

Para a conversão para este espaço de cor utiliza-se a equação 2.8:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 255 \\ 255 \\ 255 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 255 \\ 255 \\ 255 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}$$

equação 2.8 - Conversão de RGB - CMY - RGB.

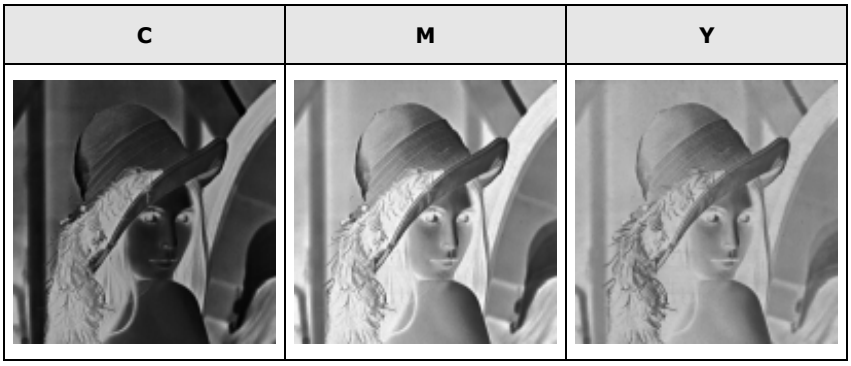


figura 2.15 - Componentes CMY.

Qualquer uma das imagens da figura 2.15 anterior é a imagem negativa das componentes RGB da figura 2.14.

Representação em YUV e YCrCb

Estes espaços baseiam-se na luminância (Y) e nas crominâncias, que são as diferenças entre os canais RGB e a luminância. Como esta diferença pode ser positiva ou negativa, e temos todo o interesse em trabalhar com números positivos deslocamos o centro das diferenças para o nível médio. Desta forma, a equação 2.4 transforma-se na equação 2.9 e a equação 2.5 na equação 2.10.

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}$$

equação 2.9 - Conversão RGB – YUV.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.168 & 0.000 & 1.602 \\ 1.168 & -0.393 & -0.816 \\ 1.168 & 2.025 & 0.000 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix} - \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}$$

equação 2.10 – Conversão YUV – RGB.

As imagens seguintes mostram as componentes YUV da imagem representada na figura 2.14

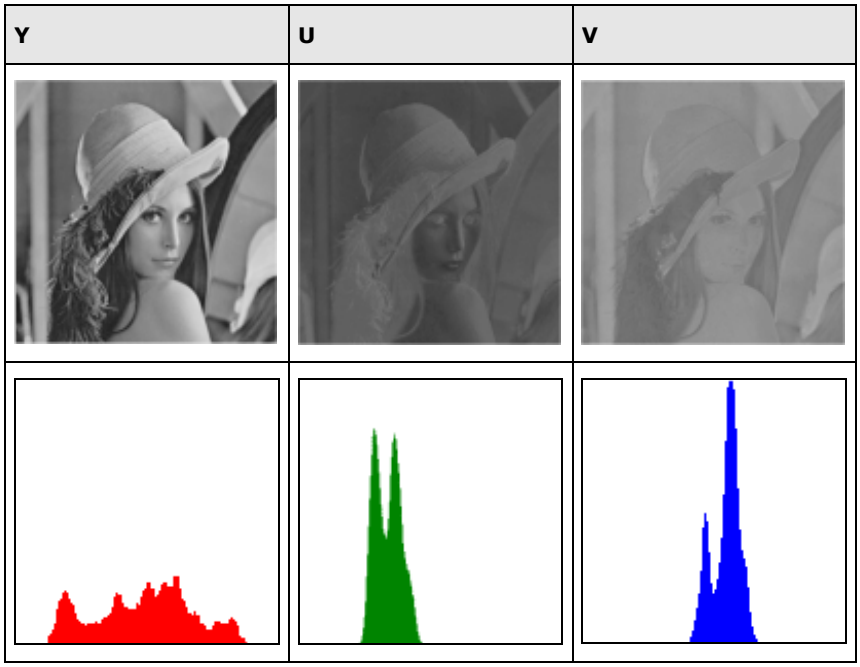


figura 2.16 – Componentes YUV e respectivos histogramas.

O espaço YCrCb destina-se às transmissões digitais e é regido pela recomendação 601-2 da CCIR[19], que estabelece os padrões internacionais de qualidade para as componentes de vídeo. O sinal da luminância é codificado com 8 bits, sendo o preto o nível 16, e o branco, o nível 232. As componentes Cr e Cb também são codificadas com 8 bites e centradas em 128.

A conversão do espaço RGB para o espaço YCrCb, descrito pela equação 2.6, e o sentido inverso, pela equação 2.7, são traduzidos pela equação 2.11 e pela equação 2.12;

[19] Comité Consultatif International des Radiocommunications

$$\begin{bmatrix} Y \\ Cr \\ Cb \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 0.255 & 0.502 & 0.097 \\ -0.147 & -0.289 & 0.437 \\ 0.437 & -0.366 & -0.071 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

equação 2.11 - Conversão RGB – YCrCb.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.168 & 0.000 & 1.602 \\ 1.168 & -0.393 & -0.816 \\ 1.168 & 2.025 & 0.000 \end{bmatrix} \begin{bmatrix} Y \\ Cr \\ Cb \end{bmatrix} - \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix}$$

equação 2.12 - Conversão YCrCb – RGB.

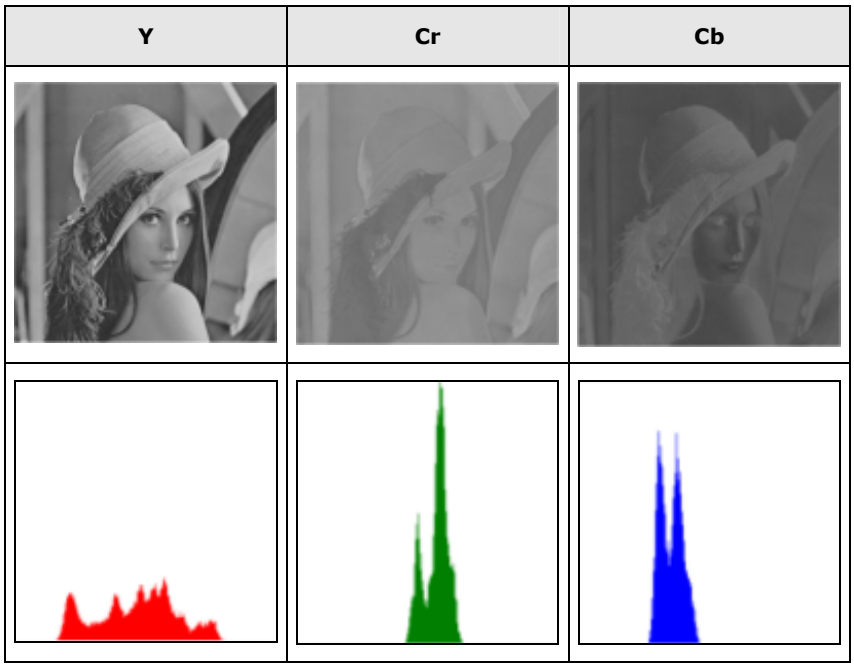


figura 2.17 - Componentes YCrCb e respectivos histogramas

Devido à grande correlação entre as componentes, as crominâncias têm uma pequena gama de valores, como se pode verificar pelos histogramas. Essa gama de valores limitada pode ser explorada pelos algoritmos de compressão.

Representação em HSL

A seguir são apresentadas as componentes do sistema HSL e os respectivos histogramas.

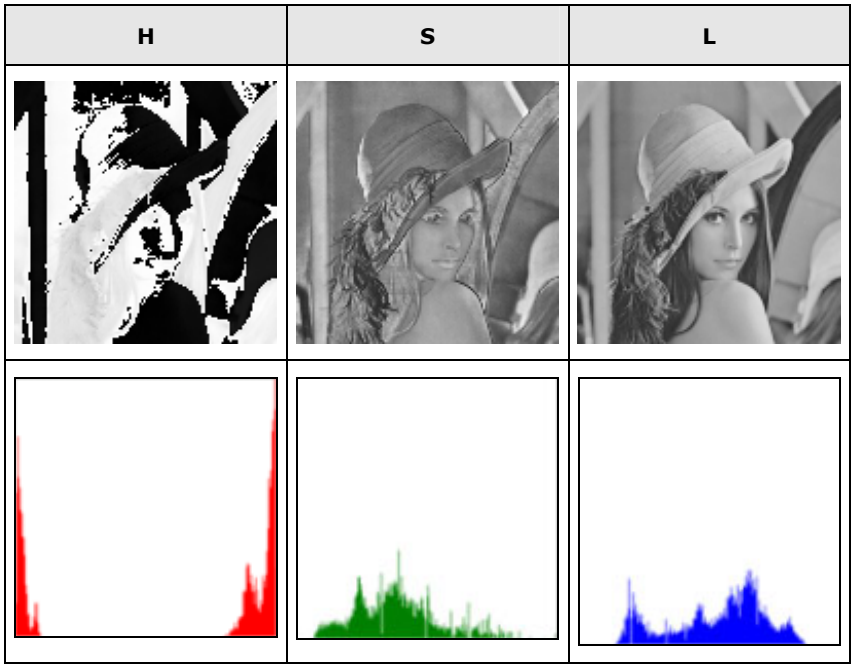


figura 2.18 - Componentes HSL e respectivos histogramas.

Neste espaço, a correlação entre os níveis mantém-se.

Representação no sistema enumerado

A figura 2.20 apresenta a paleta de cores utilizada pela figura 2.19. Cada cor da palete é acessível através de um índice entre 0 e 255, e a cada índice corresponde um valor pré-definido para cada uma das componentes RGB.

Cada cor é independente de todas as outras, não havendo, portanto, qualquer relação entre uma cor e as suas vizinhas.

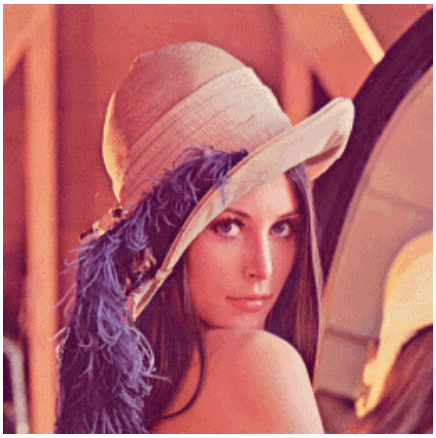


figura 2.19 – Imagem da Lena no sistema enumerado com 256 cores.

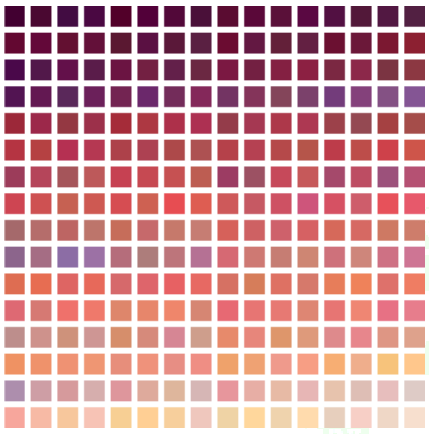


figura 2.20 - Palete de cores da figura 2.19.

Capítulo 3: Compressão de imagens coloridas com fractais

Trabalhos publicados

De entre as publicações existentes sobre a compressão de imagens com fractais, a esmagadora maioria é dedicada à compressão de imagens em tons de cinza, sendo poucos os que são dedicados à compressão de imagens coloridas. Contudo, são várias as referências à compressão de imagens coloridas, sendo a maior parte uma extensão dos métodos propostos, embora haja alguns métodos específicos para este tipo de imagens.

A grande diferença entre as imagens monocromáticas e as imagens coloridas reside no facto destas últimas estarem definidas num espaço de cor constituído por várias componentes que funcionam como um conjunto de camadas monocromáticas. A união das várias camadas monocromáticas dá a sensação de cor.

A seguir são apresentados os métodos de compressão implementados e testados neste trabalho.

Camadas separadas

A forma imediata de comprimir uma imagem com mais do que uma camada é aplicar o algoritmo a cada uma das componentes da imagem, tratando cada uma delas como uma imagem monocromática independente. Este tipo de compressão vem referido em vários trabalhos de compressão de imagens monocromáticas e é normalmente apontado como a extensão para imagens coloridas.

O ficheiro resultante contém o código das transformações de cada um das camadas da imagem original. A compressão obtida por este método é a média da compressão das camadas que compõem a imagem.

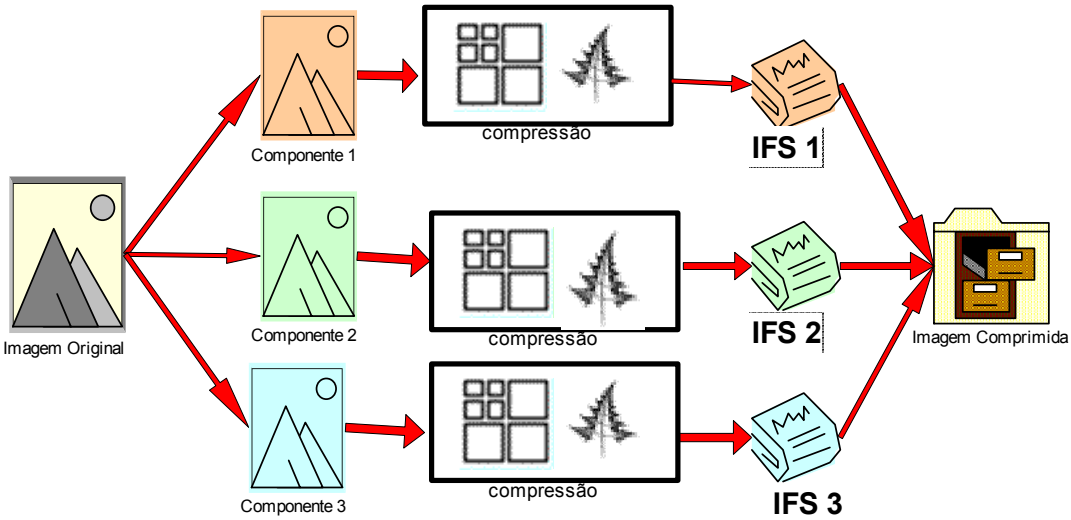


figura 3.1 – Esquema de compressão com camadas separadas.

Este método funciona em qualquer espaço de cor, tendo tantos módulos de compressão quantas as camadas que constituem a imagem.

Camadas Dependentes

Como podemos constatar no capítulo anterior, as componentes de uma imagem colorida estão mais ou menos correlacionadas conforme o espaço de cor em que se encontra.

Zhang e Po, em *Fractal Color Image Compression Using Vectror Distorcion Measure,* [ZPO95] descrevem um método para comprimir imagens coloridas. O algoritmo tira partido da grande correlação existente entre as camadas RGB e passa pela definição de transformações afim com 5 dimensões, capazes de conter o valor do factor de escala e do deslocamento para as três componentes RGB no intuito de aumentar os rácios de compressão.

$$wi = w\begin{bmatrix} x \\ y \\ zR \\ zG \\ zB \end{bmatrix} = \begin{bmatrix} a & b & 0 & 0 & 0 \\ c & d & 0 & 0 & 0 \\ 0 & 0 & sR & 0 & 0 \\ 0 & 0 & 0 & sG & 0 \\ 0 & 0 & 0 & 0 & sB \end{bmatrix}\begin{bmatrix} x \\ y \\ zR \\ zG \\ zB \end{bmatrix} + \begin{bmatrix} e \\ f \\ oR \\ oG \\ oB \end{bmatrix}$$

equação 3.1 – Tranformação afim com 5 dimensões.

Como se pode reparar na equação 3.1, existem apenas duas coordenadas na transformação[20], o que significa que o bloco definido por elas serve simultaneamente para as três camadas. Com este tipo de transformação, a compactação do IFS é mais eficiente do que nas camadas separadas, uma vez que o mesmo domínio e a mesma partição servem para as três camadas.

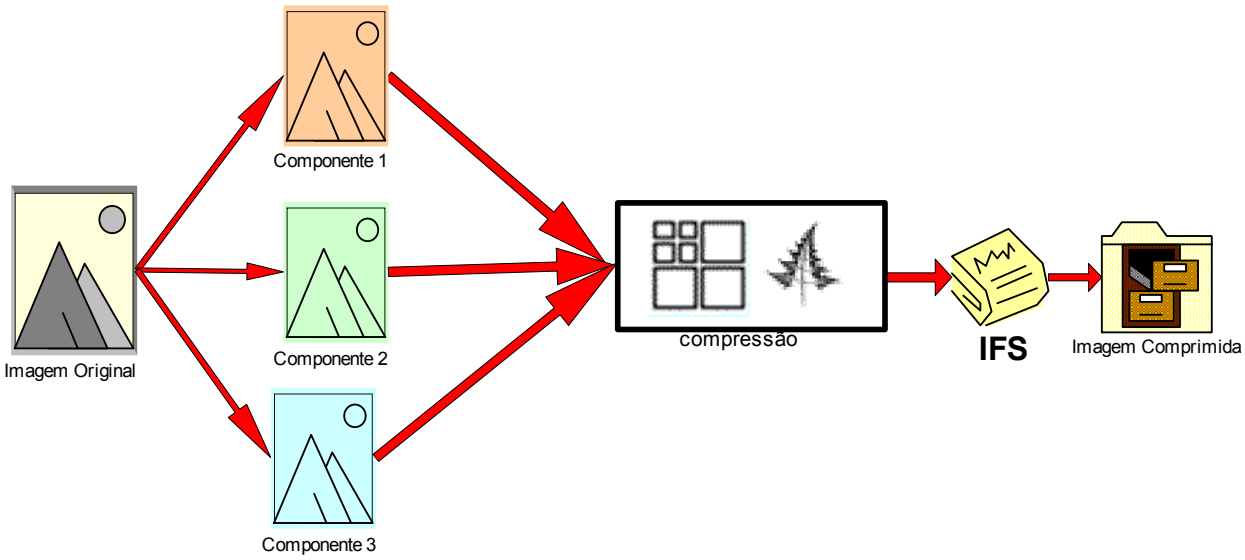


figura 3.2 - Esquema de compressão com camadas dependentes.

O algoritmo original foi definido para o espaço de cor RGB. A métrica definida para a comparação entre a região e o domínio foi a média do erro *rms* para cada uma das camadas. Esta métrica funciona bem neste espaço, porque todas as camadas contêm sensivelmente a mesma densidade de informação. Não obstante, quando se utilizam outros espaços de cor onde a densidade de informação não é a mesma, esta métrica torna-se demasiado optimista.

No nosso trabalho substituímos a média do erro pelo erro máximo. Esta métrica pode ser muito pessimista no espaço RGB, mas funciona bem nos espaços YUV e similares, uma vez que a pouca densidade de informação das camadas U e V não prejudicam a correcta codificação da camada Y.

O algoritmo começa por dividir a imagem no nível de partição mínimo (blocos maiores) e tenta satisfazer a equação 3.1 encontrando as coordenadas de um bloco do domínio cuja transformação para cada uma das camadas satisfaça a condição de erro. Caso este erro seja superior ao máximo admissível, dividem-se os blocos de cada uma das camadas em quatro novos blocos. O algoritmo continua até se atingir um erro admissível ou o nível de partição máximo ser atingido. Desta forma, a partição da imagem vai ser igual para todas as camadas.

[20] (a,c) e (b,d) - estas duas coordenadas definem um bloco.

Este método tem um tempo de execução bastante elevado, uma vez que a codificação de uma região implica uma pesquisa em todas as camadas da imagem.

Camada principal

Este algoritmo é baseado no trabalho de Hürtgen *et ai* em *Fractal Transform Coding of Color Images* [HMS94] e, tal como o anterior, explora a correlação existente entre as camadas da imagem.

Como podemos verificar capítulo anterior, nos diversos espaços de cor, as camadas de uma imagem contêm quantidades de informação diferentes, e podemos tirar partido desta característica para aumentar a velocidade de compressão. A camada G de uma imagem RGB ou a camada Y de uma imagem YUV ou a camada L de uma imagem HSL contém mais informação do que qualquer outra camada. Desta forma, podemos codificar a camada com mais informação da maneira tradicional, e as camadas restantes são submetidas ao tipo de partição das regiões da camada principal.

O método original utiliza uma tabela de códigos para determinar quais dos parâmetros da transformação afim são aproveitados para codificar as camadas secundárias. No entanto, e de acordo com os resultados apresentados, noventa por cento das transformações necessitam de um novo factor de escala e de deslocamento, enquanto que a simetria é a mesma em quase cem por cento das transformações. À luz desta análise, optamos por não utilizar a tabela e assumir que as componentes secundárias têm sempre um novo factor de escala e de deslocamento, e aproveitam a simetria, e a posição do domínio da camada principal. Esta opção permite aumentar os rácios de compressão com um pequeno sacrífico da qualidade da mesma.

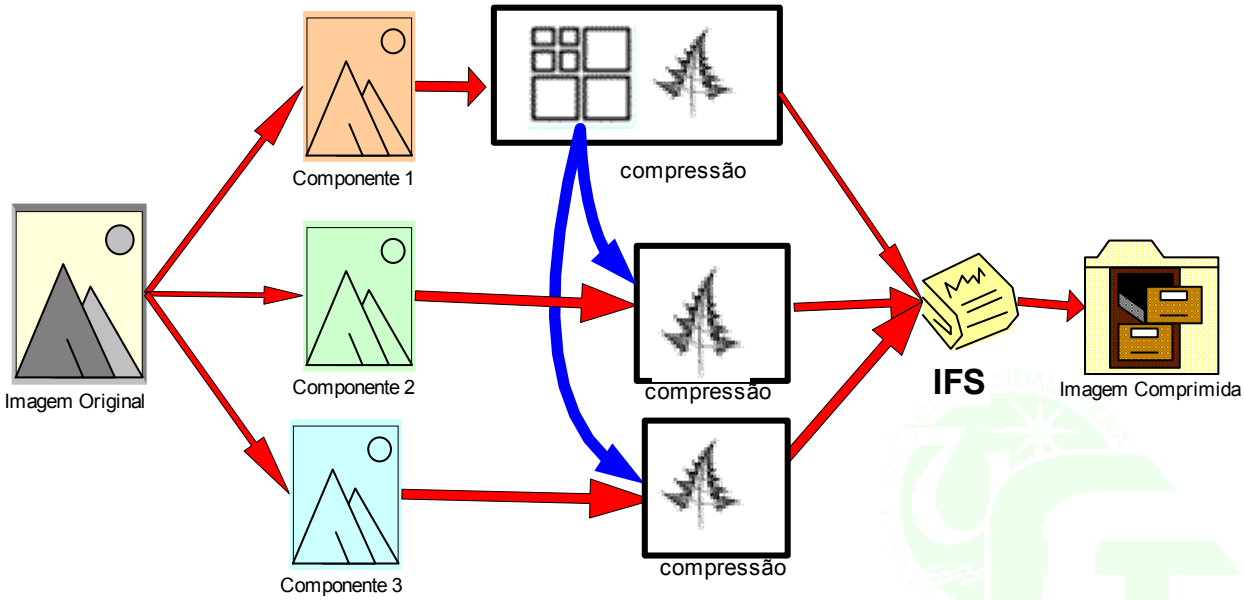


figura 3.3 - Esquema de compressão com camada principal.

As transformações afim utilizadas neste método são iguais às do método anterior, e as camadas secundárias aproveitam a partição, a simetria e a localização dos domínios da camada principal para calcularem o factor de escala e o deslocamento.

O aumento da velocidade de compressão está inerente ao facto das camadas secundárias não necessitarem de procurar a localização do domínio que codifica as suas regiões.

Camadas Reduzidas

Uma abordagem diferente para a compressão foi sugerida por Yuval Fisher em *Fractal Image Compression*- Theory and Application [FIS95]. Devido ao facto de alguns espaços de cor possuírem componentes com menos densidade de informação, estas camadas podem ser reduzidas a metade do seu tamanho original, sem que isso comprometa muito a fidelidade da compressão.

A compressão de cada uma das camadas, reduzidas ou não, pode ser feita pelo método tradicional para as camadas monocromáticas. A descodificação da imagem passa por expandir as camadas reduzidas para o seu tamanho original. Devido à falta de dimensão das transformações afim esta expansão é feita de forma automática.

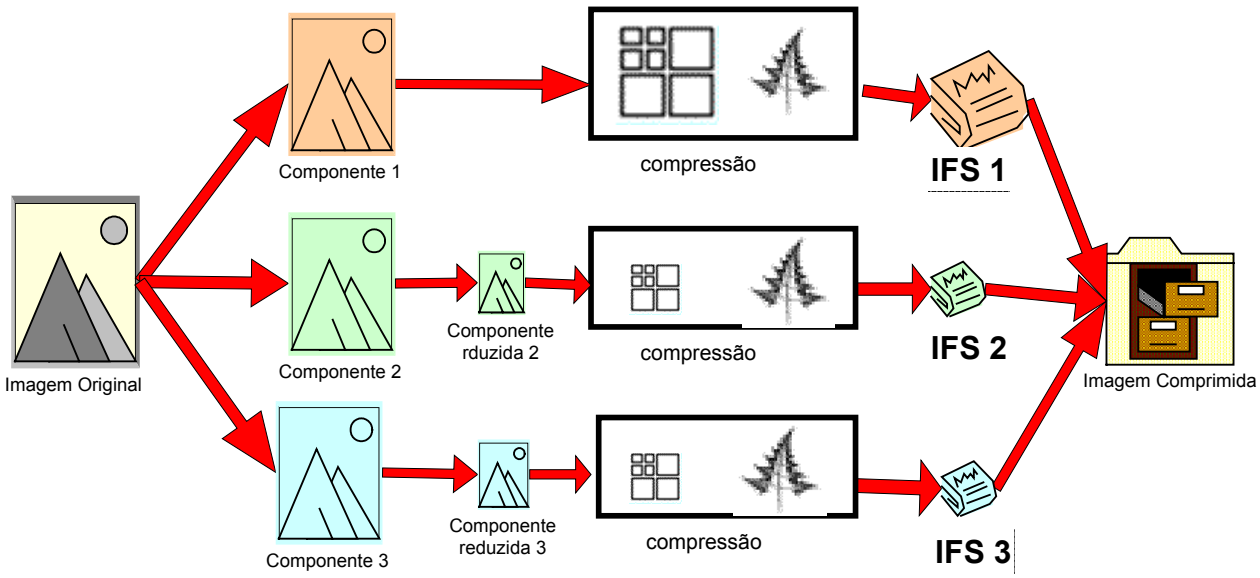


figura 3.4 – Esquema de compressão com layers reduzidos.

Em [FIS95], não foram apresentados quaisquer resultados práticos desta implementação. Em função disso, acreditamos, tendo em conta os numerosos trabalhos a que tivemos acesso, ser este trabalho o primeiro a apresentar resultados práticos dessa mesma transformação.

Compressão Híbrida

A compressão das camadas obtidas pelo método anterior pode ser obtida por qualquer outro anteriormente conhecido. Este novo método, compressão híbrida, proposto por nós visa juntar os dois métodos anteriores. As camadas das componentes com menos informação são reduzidas e, de seguida, uma das camadas funciona como principal e fornece à outra a partição, a simetria e a posição do domínio.

Com este método pretendemos aumentar a taxa e a velocidade de compressão.

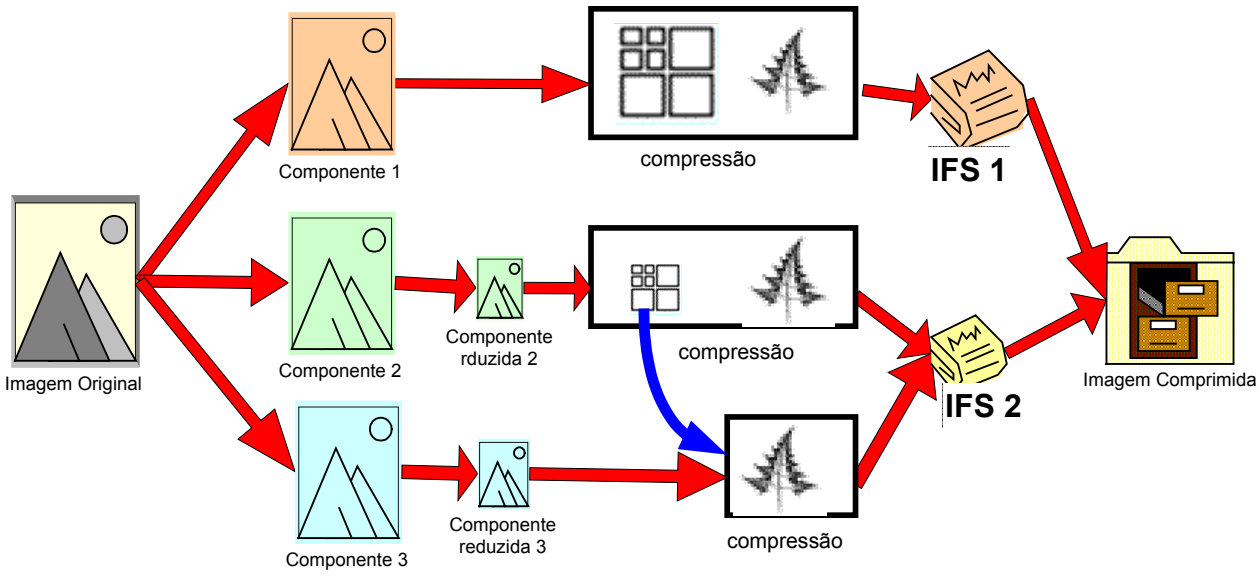


figura 3.5 - Esquema de compressão híbrido

Partilha dos domínios

Nada impede que os domínios que codificam uma região se encontrem em qualquer uma das camadas.

Nas imagens policromáticas existem várias camadas que podem ser interpretadas como um conjunto de imagens monocromáticas, mas, neste caso, cada uma destas camadas pode fornecer domínios para codificar uma região de uma camada distinta.

A união dos domínios traz um acréscimo do número de domínios, o que significa que os blocos das regiões podem ser codificados com maior fidelidade. No entanto, este acréscimo paga-se com o tempo necessário à pesquisa e com o decréscimo da taxa de compressão, uma vez que, havendo mais domínios, os bites necessários para a sua codificação são necessariamente maiores.

Métricas da compressão

De forma a podermos avaliar o desempenho dos algoritmos que nos propomos explorar, tornou-se imperioso a definição de medidas que possibilitem medir o seu desempenho. Definimos, assim, duas métricas: a fidelidade e a taxa da compressão. A primeira avalia a qualidade da compressão e a segunda mede o quanto o algoritmo consegui comprimir.

Métrica PSNR para imagens coloridas

A compressão de imagens com fractais introduz uma perda de informação durante o processo, não sendo possível a recuperação da imagem original através da descompressão. Deste modo, é de suprema importância a definição de uma medida que nos possibilite medir a qualidade da imagem comprimida em relação à imagem original. A medida mais utilizada para medir a fidelidade da compressão é a métrica PSNR.

A maneira natural de tratar a fidelidade entre a imagem original e a imagem comprimida é computar as diferenças entre as duas imagens.

Considerando *x* a imagem original com *n* pixels, e *y* a imagem obtida pelo processo de expansão da imagem comprimida, podemos definir a distância entre cada pixel pela equação 3.2:

$$d_i(x_i, y_i) = (x_i - y_i)^2 \qquad i = 0..n$$

equação 3.2 – Quadrado da diferença entre um pixel.

Em geral, são pouco úteis as diferenças individuais entre os pixels. Uma medida melhor é a média da diferença dos quadrados entre os pixels (MSE[21]):

$$MSE = \frac{1}{n}\sum_{i=0}^{n}(x_i - y_i)^2$$

equação 3.3 – MSE – (Erro quadrático médio).

Se estivermos interessados em quantificar a grandeza do erro relativamente ao sinal original podemos definir a razão entre o sinal e o ruído (SNR[22]):

$$SNR = \frac{\frac{1}{n} * \sum_{i=0}^{n}(x_i)^2}{MSE}$$

equação 3.4 – SNR – (Rácio entre o sinal e o ruído).

[21] Mean Square Error.

[22] Signal Noise Ratio.

A medida SNR pode ser expressa na escala logarítmica onde as unidades são os decibéis (dB):

$$SNR(dB) = 10 * \log_{10} \frac{\frac{1}{n} * \sum_{i=0}^{n} (x_i)^2}{MSE}$$

equação 3.5 - SNR em escala logarítmica.

A medida padrão para quantificar a fidelidade de uma imagem em relação a outra é a razão entre o sinal de pico (*p)* em vez da média dos pixels. O sinal de pico de uma imagem é o valor máximo da intensidade que os pixels podem conter. Esta medida, chamada de PNSR[23], é expressa pela equação 3.6:

$$PSNR(dB) = 10 * \log_{10} \frac{p^2}{MSE}$$

equação 3.6 – PSNR – (Razão entre o sinal e o sinal de pico).

Para as imagens a cores, o cálculo do PSNR deve ter em conta que a imagem final é a combinação de várias sub-imagens ou componentes, dependendo do espaço de cor utilizado. Estas sub-imagens podem ser tratadas como sendo independentes e, desta forma, a métrica MSE para uma imagem colorida com *k* componentes (C_1, C_2, ...,C_k) é definida por:

$$MSE_{color} = \frac{1}{K * n} \sum_{i=0}^{n} (xC_{1i} - yC_{1i})^2 + (xC_{2i} - yC_{2i})^2 + ... + (xC_{ki} - yC_{ki})^2$$

equação 3.7 – MSE para imagens coloridas.

Desta forma, o PSNR para imagens a cores é definido pela equação 35:

$$PSNR_{color}(dB) = 10 * \log_{10} \frac{p^2}{MSE_{color}}$$

equação 3.8 – PSNR para imgens coloridas.

As imagens tratadas neste trabalho são constituídas por componentes com 256 níveis de intensidade. A métrica utilizada para medir a fidelidade da compressão é expressa pela equação 3.9:

$$PSNR_{color}(dB) = 10 * \log_{10} \frac{255^2}{MSE_{color}}$$

equação 3.9 – PSNR para imagens com 255 níveis em cada componente.

[23] Peek Signal Noise Ratio.

A seguir são apresentadas imagens comprimidas da imagem Lena e as respectivas fidelidades em escala logarítmica(PSNR):

21,07	25,75	29,95	38,36

tabela 3.1- Quatro imagens da Lena com diferentes valores de PSNR.

O mínimo da escala é zero, o qual é obtido quando a diferença entre todos os pixels é o sinal de pico. O máximo é obtido pela comparação de duas imagens rigorosamente iguais, sendo o resultado um valor infinito.

Apesar da escala ser bastante alargada, imagens com valores de PSNR acima dos 35 db têm qualidade muito boa e imagens com valores de PSNR abaixo dos 15 db são imagens com um grande grau de distorção tal que as torna inutilizáveis para efeitos práticos.

O valor desta métrica depende em muito do detalhe da imagem original, pois imagens com muito detalhe fornecem valores de PSNR mais baixos do que as imagens com menos detalhes para um aspecto visual semelhante.

A vantagem de utilizar a escala logarítmica consiste na obtenção de números razoavelmente pequenos para quantificar a fidelidade. Vinte valores separam uma imagem com muito boa qualidade de uma imagem com qualidade medíocre.

Taxa de compressão

Para medir a taxa de compressão utilizamos duas medidas distintas: o rácio de compressão e o número de bites por pixel.

A primeira é a mais usual e computa o rácio entre o tamanho da imagem original com o tamanho da imagem gerada na compressão.

$$Racio = \frac{bytes\_da\_imagem\_original}{bytes\_da\_imagem\_comprimida}$$

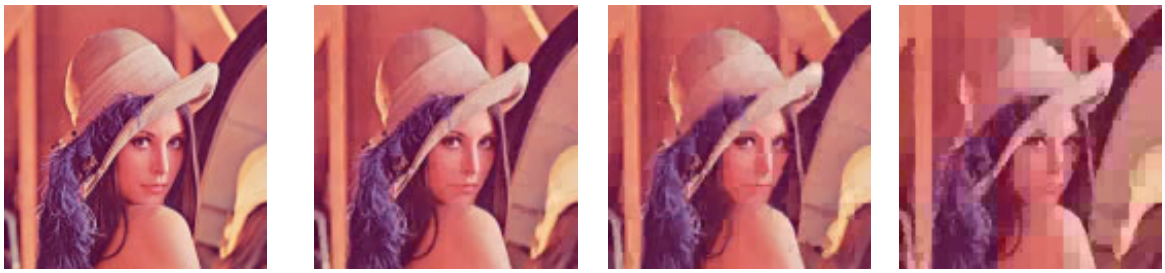
equação 3.3.10 – Cálculo do rácio de compressão

O número de bites por pixel é uma medida mais técnica e mede o número de bites que são necessários para codificar um pixel.

$$BPP = \frac{número\_de\_bits\_image\_comprimida}{número\_de\_pixels}$$

equação 3.3.11 – Cálculo da métrica bites *por pixel* (bpp)

As imagens tratadas neste trabalho possuem 24 bites para representar a cor de um pixel, sendo esta a taxa de compressão da imagem original (24 bpp).



rácio = 6	rácio = 100	rácio=200	rácio=400
bpp = 4	bpp = 0.24	bpp = 0.48	bpp = 0.06

tabela 3.2- Quatro imagens da Lena com diferentes taxas de compressão.

A tabela anterior mostra a imagem da Lena com diferentes taxas de compressão. Como se pode verificar o número de bites por pixel evolui na proporção inversa que o rácio da compressão.

Capítulo 4: Definição dos testes práticos

Escolha das Imagens

Para executar os nossos testes necessitamos de definir quais as imagens que vamos utilizar e quais os parâmetros da compressão, pois isso condiciona não só os resultados mas também a comparabilidade dos mesmos. Os critérios para a selecção das imagens basearam-se em dois factores: o uso de imagens do domínio público para que os resultados dos nossos algoritmos possam ser comparados com outros métodos e a dificuldade de compressão daquelas.

A imagem "Lena" é obrigatória, uma vez que abundam os resultados obtidos com diversos métodos de compressão e é uma imagem padrão em algoritmos de compressão. A imagem "Baboon"[24] é uma imagem de domínio público que é muito difícil de comprimir pois tem muitos detalhes e ocupa praticamente a totalidade do espaço de cor do sistema RGB. A imagem "Peppers" também é do domínio público, sendo muito utilizada em compressão de imagens. A imagem "Sky" não tem grandes detalhes e é sem dúvida, aquela que melhor se adequa a este tipo de compressão. Esta última imagem visa representar toda uma classe de imagens sobre as quais o nosso algoritmo tem particular interesse.

[24] também conhecida por "Mandrill"

figura 4.1 – Imagens utilizadas nos testes. Babbon, Peppers, Lena e Sky.

Espaço de cor

As imagens utilizadas foram adquiridas e serão apresentadas no formato RGB. Contudo, podemos converter as imagens noutro espaço de cor, comprimi-la, guardá-la e fazer a conversão inversa para apresentá-las ao utilizador. As imagens utilizadas neste trabalho são imagens de 24 bites (8 para cada componente) obtidas no formato BMP e serão apresentadas no monitor, tendo obrigatoriamente de ser convertidas para o formato RGB para a sua visualização.

Como explicámos no capítulo 2, o espaço de cor HSL é um formato que visa a simplificação do processo da escolha de uma cor, principalmente em programas de edição gráfica, e os resultados obtidos pelos métodos de compressão de imagens com fractais neste espaço de cor revelaram-se muito fracos em relação aos restantes espaços.

O espaço de cor RGB apresenta uma maior fidelidade, uma vez que não existe perda de informação com a conversão. No entanto, a taxa de compressão é menor, porque cada uma das camadas contém sensivelmente a mesma informação. Este espaço pode ter algum interesse na compressão com camadas separadas e nas camadas dependentes. Todavia os restantes métodos não se adaptam bem a este espaço uma vez que visam tirar partido da existência de uma camada com mais informação.

Com vista à uniformização dos testes sistemáticos, e com base nos testes preliminares efectuados, vamos utilizar apenas o espaço YUV que é o espaço que fornece melhores resultados práticos.

Parâmetros da compressão

A compressão de imagens coloridas com fractais tem diversos parâmetros de compressão que são visíveis na figura 4.2. A aplicação desenvolvida permite alterá-los a todos, embora esteja parametrizada com valores por omissão para a generalidade dos casos. Devido ao elevado número de parâmetros, vamos fixar alguns que se adaptam bem a este trabalho. Apresentamos a seguir uma breve explicação de cada um dos parâmetros.

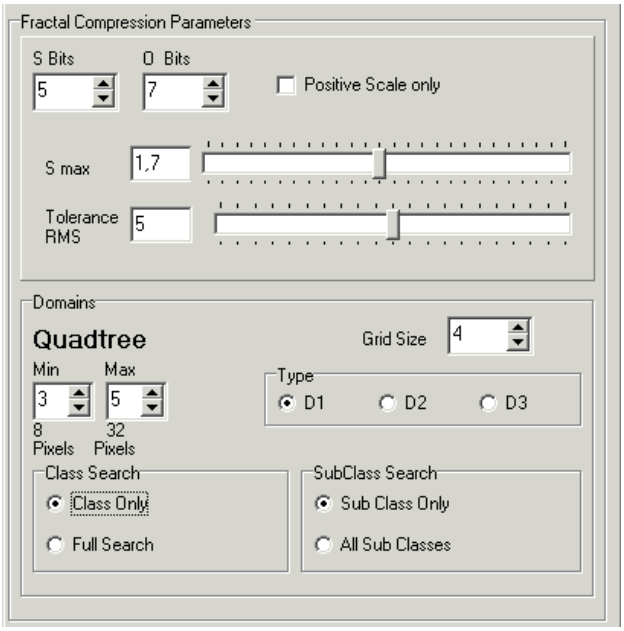


figura 4.2 - Parâmetros da compressão de imagens com fractais.

Factor de escala e deslocamento

De acordo com os resultados práticos obtidos em [FIS95], a escala codificada com cinco bites e o deslocamento codificado com sete bites fornecem uma boa relação entre fidelidade e compressão, pelo que serão usados estes valores nos nossos testes.

O valor do factor de escala pode ser negativo ou positivo, sendo que, se a escala for apenas positiva, incrementa-se a velocidade da compressão, uma vez que os domínios possuem uma classe para factores de escala negativa e outra para factores positivos. Restringindo o tipo de escala, aumenta-se a velocidade de compressão, mas reduz-se a fidelidade. Como vamos utilizar a classificação dos domínios a utilização de escalas negativas fornece-nos duas classes de pesquisa, aumentado assim a fidelidade de compressão.

Apesar do critério de convergência estrita garantir a convergência de um IFS, se a escala for menor que a unidade, a utilização de factores de escala superiores é permitida desde que o deslocamento permita repor a intensidade dentro dos níveis de cor disponíveis.

De forma a podermos definir um valor para a escala máxima, compactamos as imagens escolhidas para os testes com os mesmos parâmetros, fazendo variar apenas o valor da escala máxima e calculamos a fidelidade de cada uma. A tabela seguinte mostra o resultado destes testes:

S max	Baboon	Peppers	Lena	Sky	Média
1,00	22,33	26,88	28,19	29,15	26,64
1,20	22,40	27,04	28,32	29,30	26,76

S max	Baboon	Peppers	Lena	Sky	Média
1,40	22,44	27,07	28,33	29,35	26,80
1,60	22,45	27,08	28,36	29,40	26,82
1,80	22,45	27,09	28,35	29,50	26,85
2,00	22,46	27,12	28,33	29,50	26,85
2,20	22,46	27,10	28,35	29,46	26,84
2,40	22,46	27,14	28,32	29,42	26,84
2,60	22,45	27,13	28,27	29,38	26,81
2,80	22,44	27,11	28,27	29,29	26,78

tabela 4.1 – Variação do PSNR de várias imagens em relação ao valor máximo do factor de escala.

A próxima figura mostra, de uma forma gráfica, a média da fidelidade das imagens testadas:

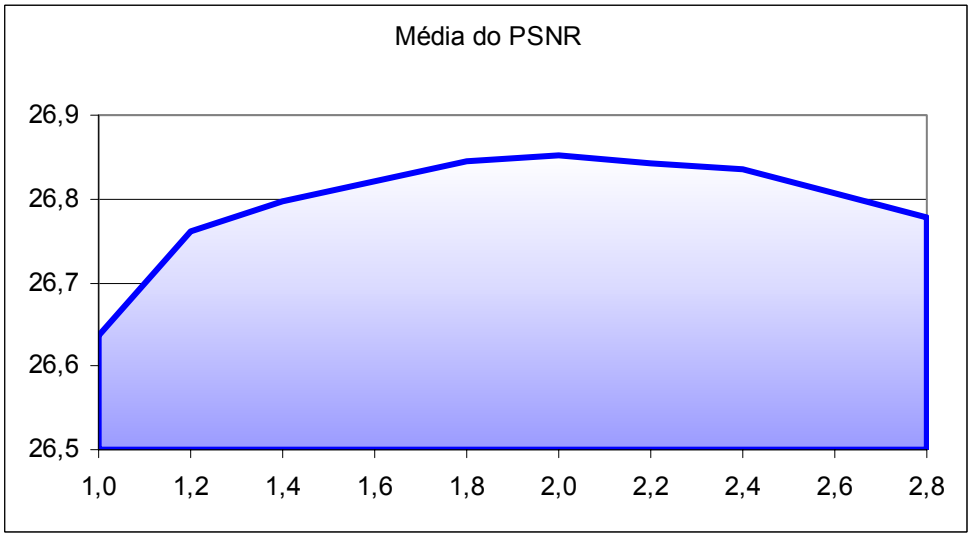


gráfico 4.1 – Gráfico da média do PSNR das imagens da tabela 4.1.

Pelo gráfico anterior podemos ver que os melhores resultados são obtidos com um factor de escala próximo de 2.0, pelo que os nossos testes vão utilizar este valor. Este factor de escala é coerente com os resultados práticos obtidos em [OMD98].

Tolerância

A tolerância RMS determina qual o erro máximo admissível entre o bloco de domínio e o bloco da região. Para valores de erro superiores ao admissível, o bloco da região é dividido em quatro. Por isso, quanto maior for o erro, menos divisões são necessárias e maior será a taxa de compressão.

Os nossos algoritmos têm, neste aspecto, uma pequena diferença em relação ao algoritmo original. Enquanto que o algoritmo original pára a pesquisa ao encontrar uma transformação com um erro inferior ao especificado, os nossos algoritmos pesquisam a

melhor transformação dentro do nível. Isto traduz-se num aumento da fidelidade da compressão sem sacrifício algum da taxa de compressão, com o único inconveniente o incremento do tempo de execução dos algoritmos.

Resultados práticos demonstram que valores de erro próximos de dez promovem uma alta taxa de compressão, enquanto que valores próximos de zero dão uma óptima fidelidade.

Domínios

Os tipos de domínios disponíveis no nosso sistema são os mesmo que foram propostos no algoritmo original.

Os domínios do tipo D1 e D2 dão uma boa taxa de compressão, porque este método produz poucos domínios e, portanto, são necessários poucos bites para a sua codificação. Os domínios de tipo D1 possibilitam uma maior fidelidade de compressão maior do que os do tipo D2 porque têm mais domínios pequenos que grandes. Vamos utilizar este tipo de domínios para codificar as imagens com uma alta taxa de compressão.

Nos domínios do tipo D3, os domínios grandes e pequenos existem em quantidades sensivelmente iguais, e será este o tipo de domínio escolhido para fazer os nossos testes de fidelidade.

A grelha dos domínios determina quantos destes serão disponibilizados para a compressão. Quantos mais existirem, maior será a fidelidade da compressão. Porém, a taxa e a velocidade de compressão evoluem na proporção inversa .

Nada impede que a pesquisa do domínio para uma região se restrinja à camada da mesma. Como as imagens coloridas são compostas por várias camadas, é natural que a codificação de uma região possa tirar partido desse facto e possa mapear uma região num domínio de qualquer uma das camadas. No entanto, se permitirmos que isso aconteça, a pesquisa vai demorar muito mais tempo, pois existem mais domínios para comparar e a taxa de compressão baixa, porque é necessário guardar a informação acerca da camada que codifica o domínio.

Como os nossos testes de fidelidade utilizam domínios do tipo D3, vamos utilizar a classificação dos domínios para limitar a pesquisa a domínios da mesma classe e da mesma subclasse, aumentando, assim, substancialmente, a velocidade de compressão.

Vamos utilizar os domínios do tipo D1 para obter uma taxa de compressão elevada, uma vez que os domínios não são sobrepostos, o que faz com que haja menos domínios e, portanto, os bites necessários para codificar um domínio são necessariamente menos. Como os domínios são poucos, vamos utilizar uma pesquisa exaustiva, prescindindo, deste modo, da classificação dos mesmos.

Ficheiros

Todos os ficheiros produzidos pelo nosso programa contêm um cabeçalho comum que especifica os parâmetros da compressão utilizados.

O cabeçalho dos ficheiro que contêm as imagens coloridas compactadas é composto por:

Campo	**Tamanho em bits**	**Valores (máximos)**
Tipo de compressão	3	8 métodos distintos
Espaço de cor	2	4 espaços de cor
Dimensão horizontal	12	4096 pixels (2^{12})
Dimensão vertical	12	4096 pixels (2^{12})
Partição mínima (Qtree)	3	partições até 256 x 256 -> $(2^2)^3$
Partição máxima (Qtree)	3	partições até 256 x 256 -> $(2^2)^3$
Tamanho da grelha	4	16 pixels
Tipo de domínios	2	D1, D2 e D3
Números de bites para a escala (s)	4	65536 valores -> $(2^2)^4$
Número de bites para os deslocamento (o)	4	65536 valores -> $(2^2)^4$
Escala máxima	5	entre 0.5 e 3.7 com intervalos de 0.1
Tipo de Compressão	3	8 métodos de compressão

tabela 4.2 – Cabeçalho do ficheiro das imagem comprimida.

Os métodos de compressão utilizados possuem os valores expressos na tabela seguinte:

Nome da Compressão	**Valor**
Camadas separadas com domínios separados	0
Camadas separadas com união de domínios	1
Camadas dependentes com domínios separados	2
Camadas dependentes com união de domínios	3
Camada principal com domínios separados	4
Camada principal com união de domínios	5
Camadas reduzidas	6
Híbrido	7

tabela 4.3 – Valores de identificação do tipo de compressão

Este cabeçalho permite guardar a informação necessária para descomprimir a imagem compactada com todos os parâmetros utilizados nesta investigação.

A restante informação do ficheiro contém a partição da imagem e a codificação das transformações afim. Para codificar a partição da imagem, foi usado o método quadtree, que necessita apenas de um bit para indicar a divisão ou não de uma região.

Cada transformação é codificada com os seguintes parâmetros:

Campo	Tamanho
Escala (s)	Definido no cabeçalho
Deslocamento (o)	Definido no cabeçalho
Tipo de simetria	3 bits
Número do domínio	Dependente do tipo e da grelha e do tipo de domínio.
Camada do domínio	2 bites [25]

tabela 4.4 – Codificação de uma transformação no ficheiro de imagem comprimida

A forma como as transformações estão armazenadas no ficheiro depende do método de codificação utilizado.

Configurações pré-definidas

O nosso trabalho permite comprimir em quantidade ou qualidade, sendo estas mutuamente exclusivas. Uma vez que no mercado existem já boas soluções para resolver a questão da qualidade, vamos programar o nosso algoritmo para a resolução da quantidade, pois é aí que se situa o maior vazio de soluções.

Os parâmetros a seguir enunciados estão optimizados para imagens de 512 por 512 pixels. Para imagens com dimensões diferentes, devem afinar-se os valores do tamanho da grelha e da quadtree.

Para a elaboração dos nossos testes vamos utilizar quatro configurações distintas: fidelidade, médio, compressão e extremo.

Fidelidade

Os resultados obtidos com esta configuração visam uma maior fidelidade da compressão. A melhor qualidade das imagens assim obtidas tem por contraponto um decréscimo da taxa de compressão.

Neste tipo de teste foram usados os seguintes valores dos parâmetros:

Campo	Valor
Bits da escala	5
Bits do deslocamento	7

[25] Apenas quando a codificação utilizar a união dos domínios.

Campo	Valor
Escala máxima	2.0
Quadtree (Min – Max)	3 –5 [26]
Grelha	4
Tipo de Domínio	D3
Pesquisa	Apenas na mesma classe e na mesma subclasse
Erro máximo admissível (RMS)	5.0

tabela 4.5 – Parâmetros de compressão com ênfase na fidelidade

A fidelidade obtida com estes parâmetros é aquela que consideramos razoável em relação ao tempo de compressão. A pesquisa exaustiva do domínio dá uma fidelidade maior sem comprometer a taxa de compressão. Porém, o tempo gasto na compressão compromete seriamente a sua viabilidade prática.

Os nossos algoritmos permitem a obtenção de imagens comprimidas com uma fidelidade muito maior do que a apresentada nesta configuração, mas o tempo necessário para a sua obtenção é inversamente proporcional à taxa de compressão obtida, tornando os resultados medíocres quando comparados com outros algoritmos. Esta configuração pretende marcar o início do intervalo de compressão a partir do qual o nosso algoritmo se torna interessante e manifesta as suas melhores qualidades.

Médio

Os resultados obtidos com esta configuração visam o teste de compromisso entre a fidelidade e a taxa de compressão. Neste tipo de testes utilizamos os mesmos parâmetros usados no teste anterior aumentando o erro máximo admissível para 7.5. Com este aumento do erro espera-se que alguns blocos que foram divididos no método anterior o não sejam agora, aumentando, assim, a taxa de compressão e, de modo inverso, o tempo de compressão e a fidelidade.

Compressão

Os resultados obtidos com esta configuração visam a obtenção de taxas de compressão elevadas, sacrificando a fidelidade da compressão.

Neste tipo de teste foram implementados os parâmetros abaixo indicados:

Campo	Valor
Bits da escala	7
Bits do deslocamento	5

[26] Partição da imagem em blocos com dimensões 8x8, 16x16 e 32x32 pixels

Campo	Valor
Escala máxima	2.0
Quadtree (Min – Max)	4 –6 [27]
Grelha	1
Tipo de Domínio	D1
Pesquisa	todos os domínios
Erro máximo admissível (RMS)	5.0

tabela 4.6 - Parâmetros de compressão com ênfase na compressão

A taxa de compressão é obtida através do aumento da dimensão dos blocos de partição. Os blocos mais pequenos vão ter as dimensões de 16x16, que são quatro vezes maiores do que os dos métodos anteriores.

A pesquisa exaustiva dos domínios não compromete o tempo de compressão, porque o tipo de domínio escolhido gera poucos domínios.

Extremo

Esta configuração utiliza os mesmos parâmetros da configuração anterior e visa dar uma ideia dos limites da compressão de imagens com fractais. Este tipo de compressão gera imagem com bastante degradação. Contudo, consegue taxas de compressão muito acima da média.

Estes dois últimos tipos de configurações são adequados quando se pretendem imagens cujos detalhes são menos importantes do que o tamanho que essas mesmas imagens ocupam.

Domínios em cada configuração pré-definida

A tabela seguinte contém o número de domínios existentes em cada uma das camadas para cada uma das configurações. Como se pode constatar, as configurações "*fidelidade*" e "*médio*" contêm um número de domínios suficientemente grande para poderem ser classificados, dando em média cerca de duzentos elementos para cada subclasse. A desvantagem da existência de tantos domínios reside no facto de cada um destes ter de ser classificado, o que leva tempo, e para guardar a seu endereço são necessários mais bits. A classificação dos domínios é imperiosa pois a comparação com tão elevado número de blocos torna o tempo de compressão inaceitável.

[27] Partição da imagem em blocos com dimensões 16x16, 32x32 e 64x64 pixels.

Os modos de *compressão* e *extremo* possuem poucos domínios, sendo, portanto, a classificação dispensável e até desaconselhável.

	Nivel 1	Nivel 2	Nivel 3
Fidelidade	15.876	15.376	14.400
Médio	15.876	15.376	14.400
Compressão	1.024	256	64
Extremo	1.024	256	64

tabela 4.7 - Número de domínios em cada nível da partição qadtree.

Para codificar cada um dos domínios expressos na tabela anterior, são necessários os bites assinalados na tabela seguinte:

	Nivel 1	Nivel 2	Nivel 3
Fidelidade	14	14	14
Médio	14	14	14
Compressão	10	8	6
Extremo	10	8	6

tabela 4.8 – Número de bites para a codificação de um domínio.

Comparação dos Resultados

Os resultados obtidos nos nossos testes adquirem um novo significado quando comparados com outros métodos existentes. Para esta comparação utilizamos os resultados obtidos pelo algoritmo JPEG, que é um padrão na indústria de compressão de imagens e que está disponível na linguagem de programação por nós usada, possibilitando-nos, assim, a obtenção dos resultados da compressão feita com as imagens escolhidas para os testes.

A compressão de imagens com o algoritmo JPEG tem uma melhor aplicabilidade em compressões com qualidade e obtém excelentes resultados neste campo, mas estando limitada à taxa de compressão. O algoritmo permite efectuar compressões com uma qualidade aceitável até uma taxa a rondar a centena. A partir deste valor, a qualidade de compressão baixa drasticamente e, nos testes efectuados, a própria taxa de compressão máxima limita-se a cerca de uma centena e meia.

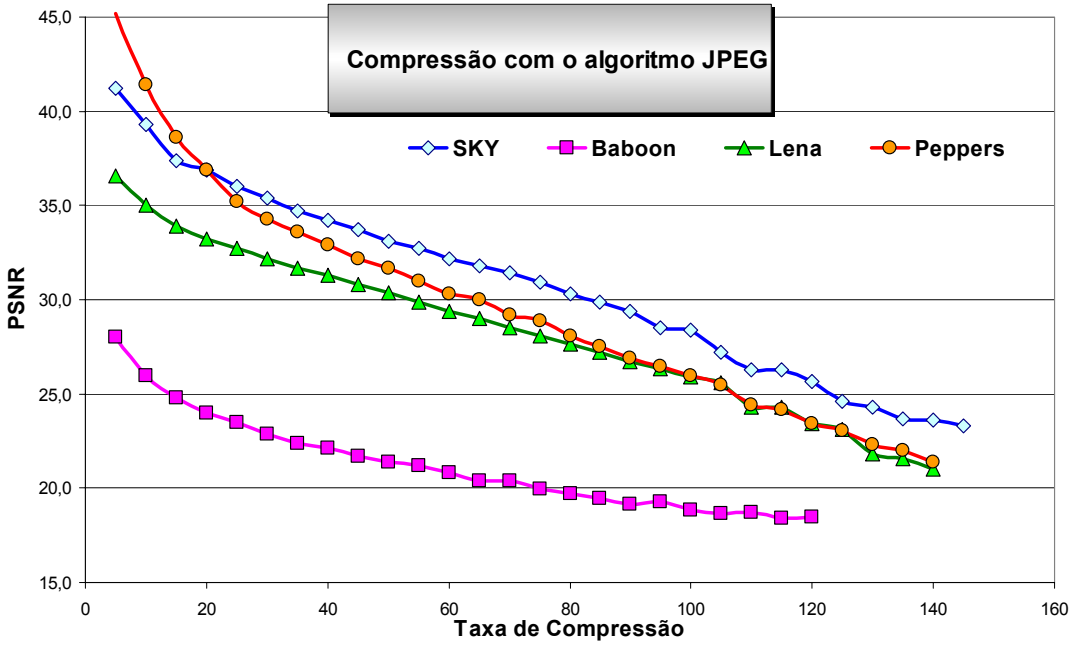


gráfico 4.2 – Compressão JPEG - Variação da fidelidade em relação à taxa de compressão.

.

Capítulo 5: Resultados práticos dos algoritmos

A seguir são apresentados os resultados da implementação dos métodos descritos no capítulo anterior. Os algoritmos são baseados no código fornecido por Yuval Fisher em [FIS95] procedendo às devidas alterações para comprimir imagens coloridas. O código foi convertido para classes de modo a obtermos uma melhor integração no ambiente de programação escolhido.

Os restantes algoritmos foram codificados em C++ e implementados no ambiente de programação Borland C++ Builder 5.0. Os resultados práticos foram obtidos num computador com processador AMD athlon 700Mhz e 256 Mb de memória RAM.

Compressão de imagens através de camadas separadas

Este é o método mais imediato para a codificação de imagens coloridas tendo como base um algoritmo que codifica imagens em tons de cinza, como é o caso do algoritmo fornecido por Yuval Fisher.

Aplica-se o algoritmo às três camadas da imagem original, como se se tratasse de três imagens distintas. Cada camada sofre uma partição autónoma, da qual são calculados os coeficientes das transformações afim que comprimem a imagem. As transformações são guardadas no ficheiro camada por camada em conjunto com a partição da imagem.

Este tipo de compressão possibilita que a pesquisa do bloco do domínio seja feita em qualquer uma das camadas que compõem a imagem. Neste caso, as transformações afim necessitam de mais dois bites para armazenar a camada que contém o domínio.

Para além dos bites indispensáveis à codificação do domínio de uma transformação, é necessária ainda a codificação dos restantes parâmetros.

A tabela seguinte indica a quantidade de bites que cada transformação afim ocupa, com excepção da codificação do domínio, que está expressa na tabela 4.8:

Campo	**Número de bits**	
	Domínios separados	**União de domínios**
Escala (s)	5	5
Offset (o)	7	7
Simetria	3	3
Camada do domínio	0	2
Total	**15**	**17**

tabela 5.1 - Número de bites de uma transformação afim no método das camadas separadas sem a codificação do domínio.

O gráfico seguinte mostra o tamanho em bites das transformações afim com domínios separados.

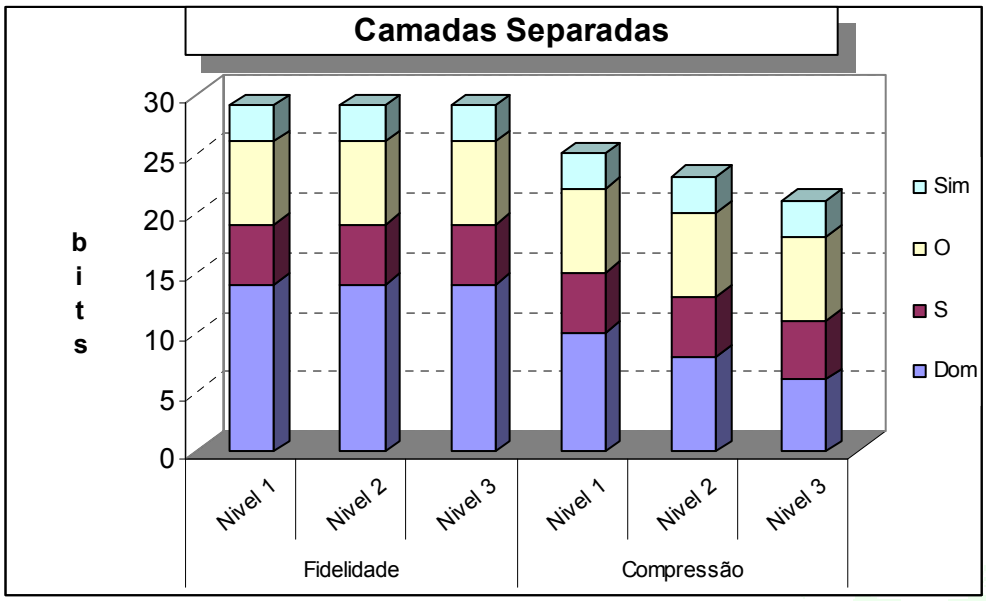


gráfico 5.1 – Tamanho em bites das transformações afins do método de camadas separadas com domínios separados dos três níveis da partição quadtree.

Domínios separados

A seguir são apresentados os resultados práticos da compressão das imagens escolhidas, com o método de camadas separadas e com domínios separados.

O tempo está expresso em segundos, o PSNR está expresso em decibéis e a taxa de compressão obtida pelo rácio entre o tamanho da imagem original e o tamanho da imagem comprimida vem expressa pelo termo "vezes".

Tempo	**Baboon**	**Lena**	**Peppers**	**Sky**	**Média**
Fidelidade	78	54	54	45	57,8
Médio	20	13	14	13	15,0
Compressão	35	16	17	9	19,3
Extremo	20	8	10	5	10,8

Compressão	**Baboon**	**Lena**	**Peppers**	**Sky**	**Média**
Fidelidade	18	51	54	84	51,8
Médio	25	81	81	117	76,0
Compressão	88	148	133	273	160,5
Extremo	135	298	224	436	273,3

PSNR	**Baboon**	**Lena**	**Peppers**	**Sky**	**Média**
Fidelidade	21,2	29,8	31,0	29,3	27,8
Médio	20,8	28,4	29,2	28,1	26,6
Compressão	19,5	25,9	26,4	25,2	24,3
Extremo	19,3	24,3	24,3	24,3	23,1

tabela 5.2 - Resultados práticos do algoritmos de camadas separadas com domínios separados.

A seguir são apresentados os gráficos da relação entre a taxa e a fidelidade de compressão de cada uma das imagens com o nosso algoritmo e com o algoritmo JPEG.

As linhas do gráfico estão definidas entre o mínimo e o máximo obtidos pelos resultados práticos dos dois algoritmos.

No caso do nosso algoritmo, os valores intermédios foram obtidos por interpolação exponencial dos valores obtidos experimentalmente.

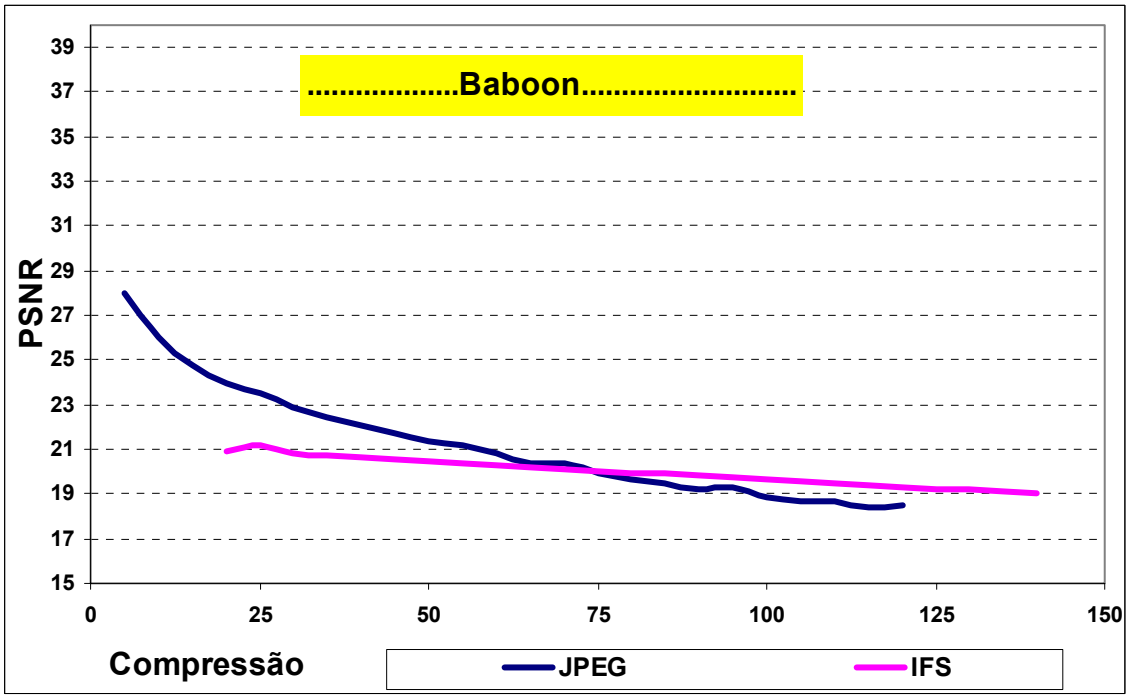


gráfico 5.2 – Resultados práticos da imagem *Baboon* com o método das camadas separadas e domínios separados.

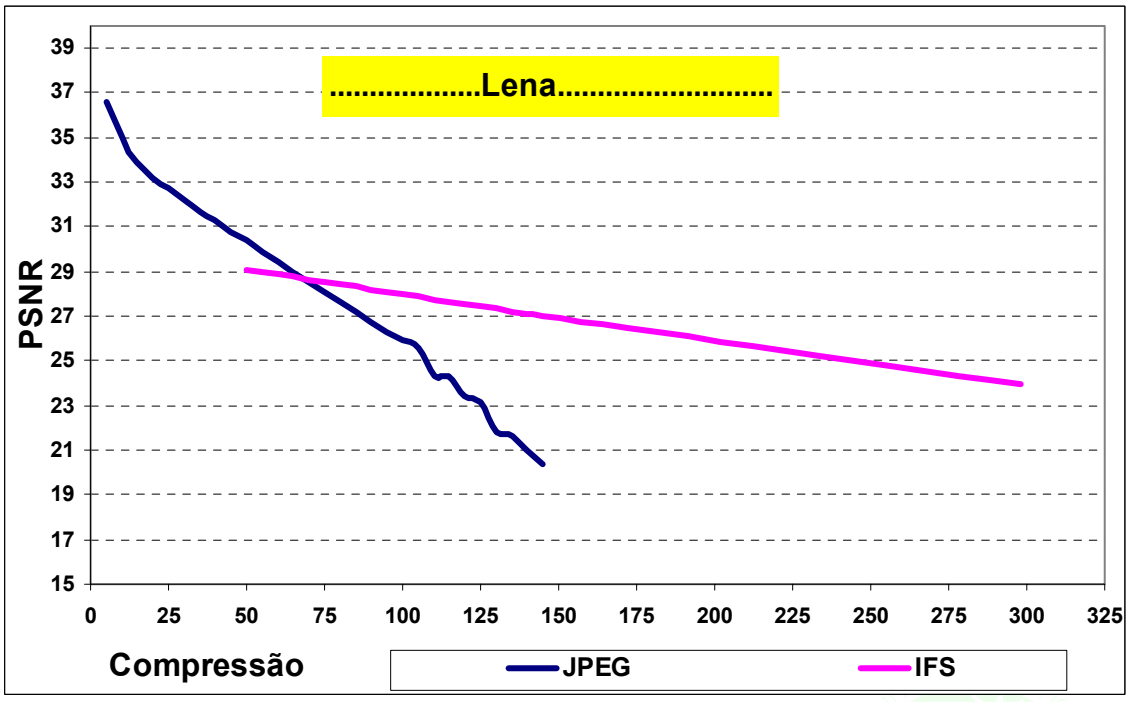


gráfico 5.3 – Resultados práticos da imagem *Lena* com o método das camadas separadas e domínios separados.

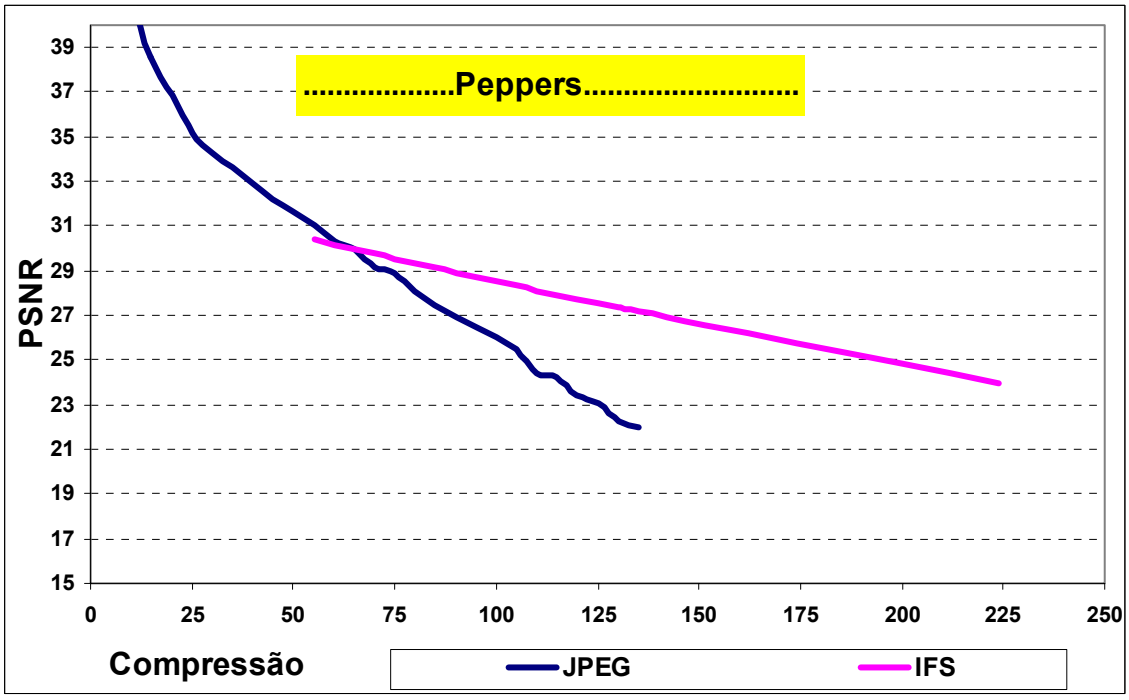


gráfico 5.4 – Resultados práticos da imagem *Peppers* com o método das camadas separadas e domínios separados.

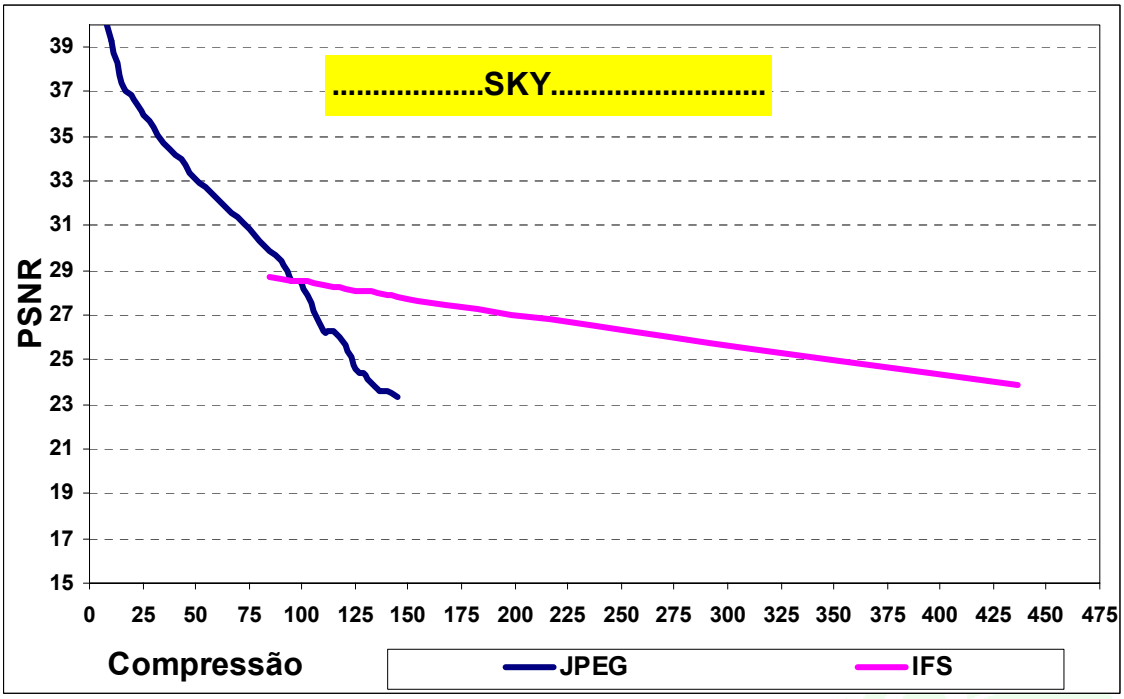


gráfico 5.5 - Resultados práticos da imagem *Sky* com o método das camadas separadas e domínios separados.

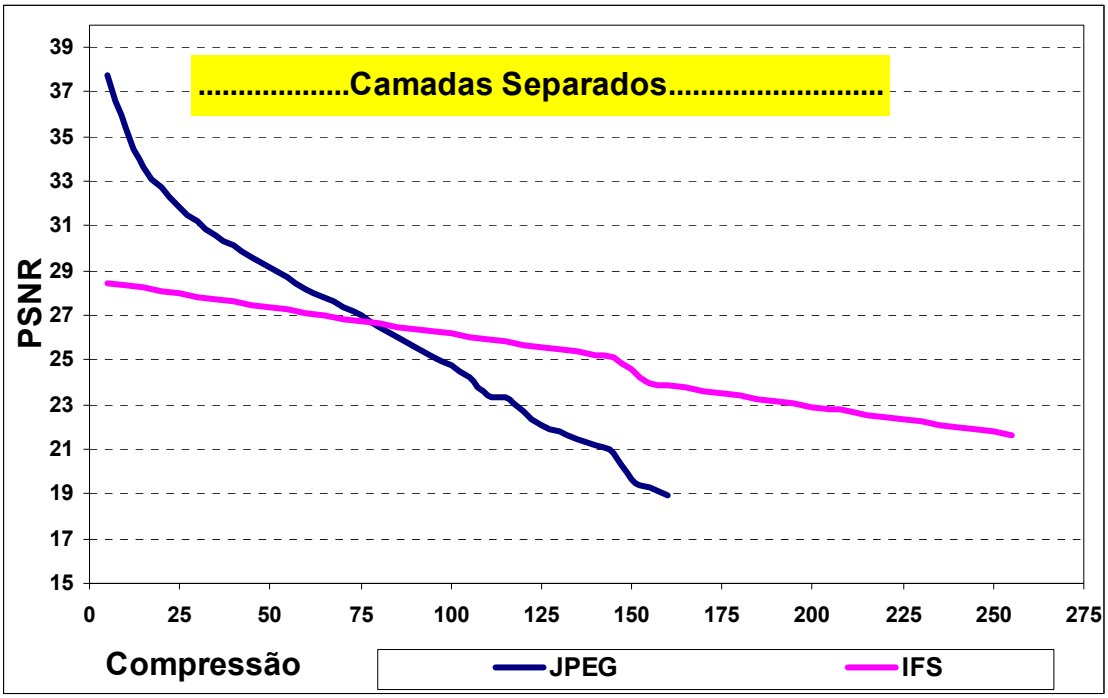


gráfico 5.6 – Média dos resultados práticos com o método das camadas separadas e domínios separados.

O gráfico anterior inclui a média de PSNR das quatro imagens em relação à taxa de compressão. Uma vez que cada uma das imagens tem um máximo distinto, optou-se por elaborar um gráfico com compressões até 250 vezes. Os valores em falta para cada uma das imagens foram obtidos por projecção dos valores de PSNR entre uma taxa de compressão entre 5 e 250 vezes para o IFS, e 5 e 160 vezes para o JPEG.

O processo de construção dos gráficos repetiu-se nos restantes métodos.

Crítica dos resultados

Em qualquer uma das imagens comprimidas, o algoritmo JPEG é melhor para taxas de compressão baixas, sendo o nosso algoritmo melhor para altas taxas de compressão.

A imagem *Baboon* é uma imagem com muitos detalhes e, por conseguinte, muito difícil de comprimir. O nosso algoritmo pode comprimir um pouco mais do que o algoritmo JPEG e obtém uma maior fidelidade em grandes taxas de compressão. Este elevado nível de detalhe leva a que a partição seja na maioria dos blocos levado até ao último nível, o que faz com que a taxa de compressão seja baixa e o tempo de compressão elevado.

As imagens *Lena* e *Peppers* são duas imagens equilibradas em termos de detalhes. Por isso, o nosso algoritmo consegue comprimir blocos das camadas superiores da partição, o que faz com que a taxa de compressão suba e supere em muito a taxa

máxima de compressão do algoritmo JPEG. Em termos de fidelidade, o algoritmo fornece resultados satisfatórios em relação à taxa de compressão.

A imagem *Sky* é uma imagem sem grande detalhe. Assim existe uma probabilidade significativa de codificar os blocos com os primeiros níveis da partição, possibilitando uma economia de tempo na pesquisa do domínio e uma boa taxa de compressão, comparativamente ao algoritmo JPEG.

O nosso algoritmo no modo de *fidelidade* leva em média um minuto a comprimir uma imagem e obtém um PSNR de apenas 28 db. Para este tipo de compressão, o algoritmo JPEG é muito melhor, pois a codificação é imediata e a sua fidelidade é muito maior em qualquer uma das imagens.

No modo de *Médio,* os resultados começam a ser mais animadores, pois, como aumentamos o erro admissível, a pesquisa de um bloco é mais rápida e pode ser codificada num nível mais elevado, o que se traduz também numa economia de espaço. Todas as imagens apresentam melhores resultados do que o algoritmo JPEG, com excepção da imagem *Baboon.*

Os modos *compressão* e *extremo* não têm comparação com o algoritmo JPEG, pois as suas taxas de compressão são muito mais elevadas. A sua codificação é rápida, mas a qualidade das imagens comprimidas é reduzida, o que se traduz numa baixa fidelidade.

A utilização do espaço de cor YUV vem beneficiar em muito o desempenho do algoritmo, pois, com a existência de duas camadas com uma baixo teor de informação, a codificação de ambas é feita com os níveis superiores da partição.

Este método também se aplica a imagens no espaço RGB, e obtêm-se imagens com mais qualidade, pois não existe perda de informação na conversão. Como todas as camadas contêm sensivelmente a mesma quantidade de informação, as camadas são codificadas com uma partição semelhante, o que reduz a taxa de compressão e aumenta o tempo de compressão, tornando este espaço de cor menos atraente face ao escolhido.

União dos Domínios

Tal como atrás foi enunciado, a possibilidade da união dos domínios é uma realidade. Essa união dos domínios faz com que o espaço de procura seja aumentado fazendo com que o bloco de uma região seja codificado com um domínio de uma camada distinta, como ilustra a figura 5.1:

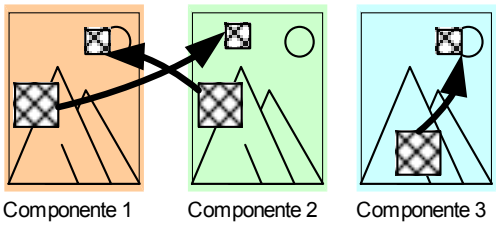


figura 5.1 – codificação de três regiões com o método de camadas separadas e domínios unidos.

A seguir são apresentados os resultados práticos do nosso algoritmo através da união dos domínios:

Tempo	Baboon	Lena	Peppers	Sky	Média
Fidelidade	238	148	159	126	167,8
Médio	55	33	36	32	39,0
Compressão	115	49	52	27	60,8
Extremo	61	23	29	16	32,3

Compressão	Baboon	Lena	Peppers	Sky	Média
Fidelidade	17	49	51	81	49,5
Médio	24	76	77	111	72,0
Compressão	81	138	126	257	150,5
Extremo	125	280	212	407	256,0

PSNR	Baboon	Lena	Peppers	Sky	Média
Fidelidade	21,2	29,8	30,8	29,3	27,8
Médio	20,8	28,4	29,0	28,2	26,6
Compressão	19,5	25,9	26,4	25,4	24,3
Extremo	19,3	23,7	23,7	24,4	22,8

tabela 5.3 – Resultados práticos do algoritmos de camadas separadas com domínios unidos.

Crítica dos resultados

Como já seria de esperar, o tempo de compressão aumentou para sensivelmente o triplo, pois como existem três vezes mais domínios a pesquisa demora muito mais. No entanto, a qualidade das imagens manteve-se e, nalguns casos diminuiu. Este facto

explica-se pela partição utilizada para a codificação da imagem. Como agora existem muitos mais domínios, a probabilidade de codificar uma região num nível superior da quadtree é mais elevada, e existem alguns blocos que conseguem tal feito. Isto aumenta a taxa de compressão, mas aumenta também o erro de compressão, pois os blocos das regiões são codificados com o melhor bloco do nível.

A taxa de compressão diminui tendo em conta que cada transformação tem agora mais dois bits, e a economia de espaço conseguida pela codificação dos blocos nos níveis superiores da quadtree não consegue compensar os bites adicionais.

Comparação entre domínios separados e domínios unidos

A união dos domínios neste método apenas veio piorar o desempenho do algoritmo. Este algoritmo é lento, e o acréscimo de tempo introduzido é inaceitável. Apenas uma imagem obteve melhores resultados em termos de fidelidade, sendo estes marginais e facilmente deturpados pelo abaixamento da taxa de compressão.

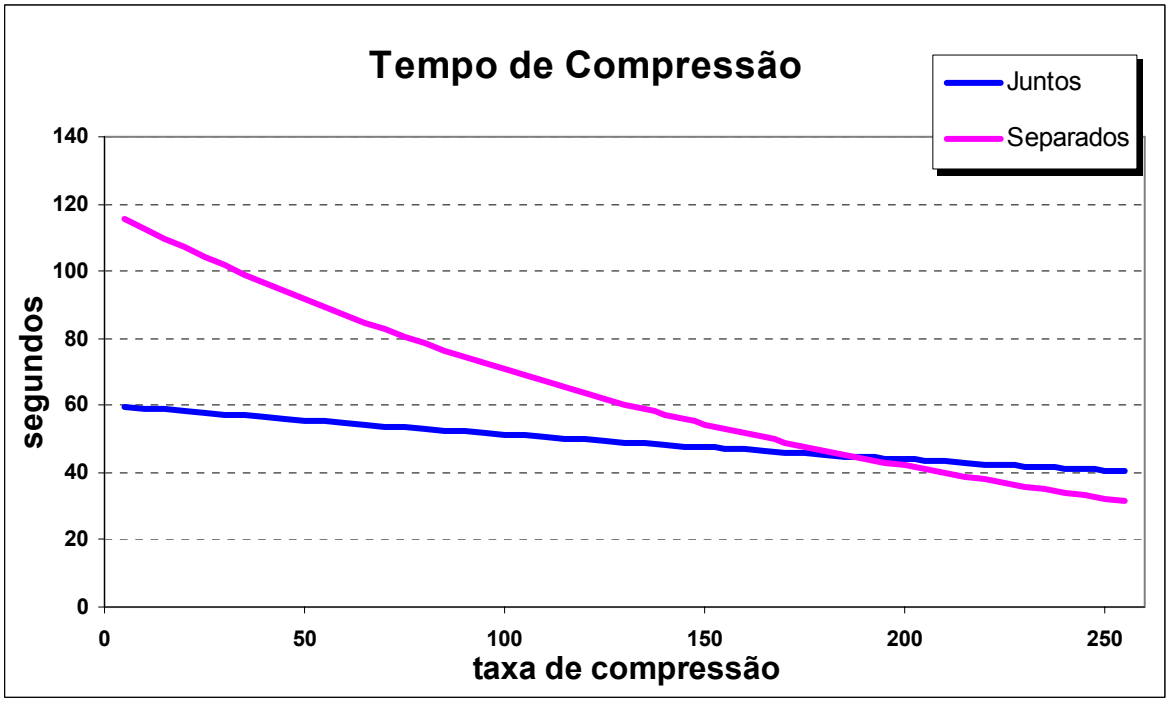


gráfico 5.7 – Tempo de compressão (segundos) do método de camadas separados com domínios separados e unidos.

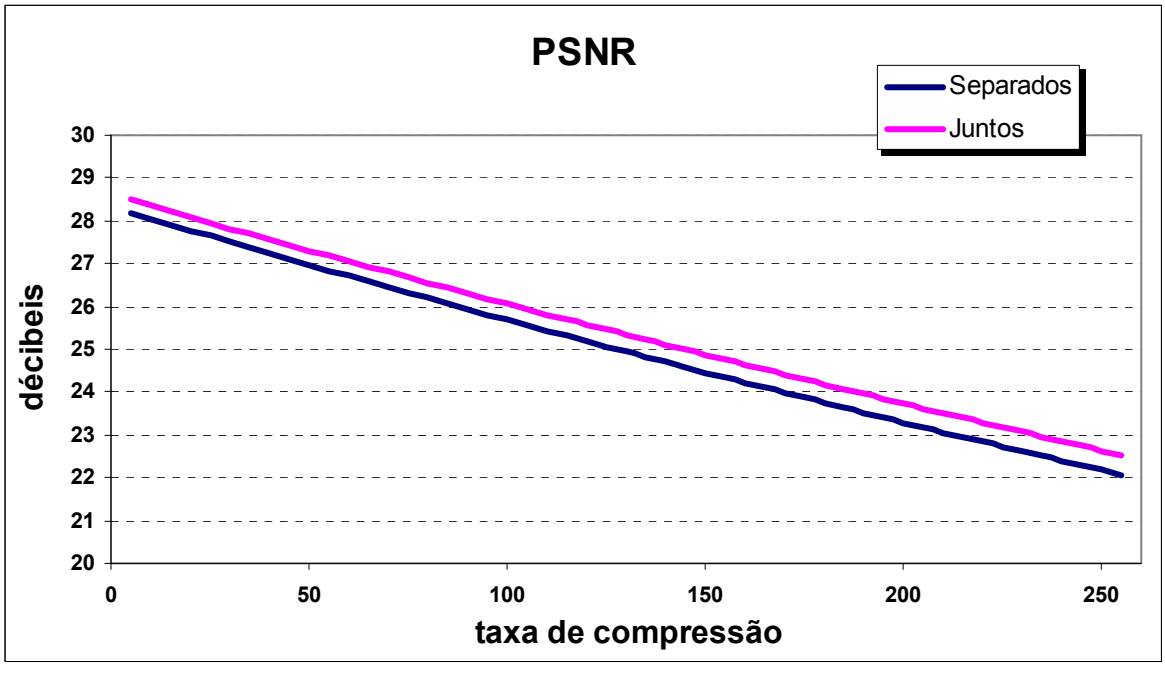


gráfico 5.8 - Fidelidade da compressão (db) do método de camadas separados com domínios separados e unidos.

Compressão de imagens através de camadas dependentes

Como se pode constatar através da observação do tamanho de cada transformação afim utilizada no método anterior, a codificação do domínio ocupa sensivelmente metade dos bites da transformação. Este método visa minimizar o espaço para a codificação do domínio.

A forma implementada por este método é a codificação de um domínio que sirva simultaneamente para as três camadas, conforme a figura 5.2. Desta forma, vai existir apenas um IFS para codificar simultaneamente as três camadas. Este IFS vai conter informação adicional, como está expresso na equação 3.1.

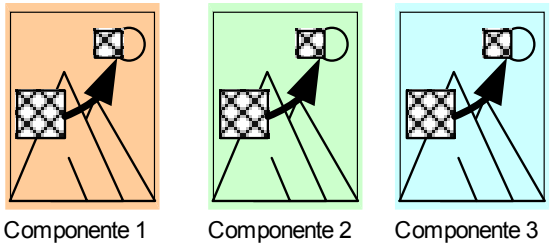


figura 5.2 – Codificação de três regiões com o método de camadas separadas e domínios separados.

Uma vez que a partição é a mesma para cada camada, é preciso guardar apenas uma partição; como o endereço dos domínios é igual para as três camadas, só é necessário guardar também um valor. São estes dois factores que fazem com que o tamanho de um IFS deste tipo seja muito menor relativamente aos respectivos três IFS normais.

A tabela seguinte apresenta o tamanho em bites de cada transformação afim, com excepção do domínio:

Campo	Número de bits	
	Domínios separados	União de domínios
Escala (s)	5 * 3	5*3
Offset (o)	7 *3	7*3
Simetria	3	3
Camada do domínio	0	2*3
Total	39	45

tabela 5.4 - Número de bites de um transformação IFS no método das camadas separadas sem a codificação.

A seguir são apresentados os resultados práticos obtidos por este algoritmo com a opção de domínios separados:

Tempo	Baboon	Lena	Peppers	Sky	Média
Fidelidade	151	112	117	62	110,5
Médio	38	23	25	14	25,0
Compressão	71	58	57	30	54,0
Extremo	66	40	38	21	41,3

Compressão	Baboon	Lena	Peppers	Sky	Média
Fidelidade	29	50	43	82	51,0
Médio	31	57	57	102	61,8
Compressão	128	146	141	246	165,3
Extremo	128	176	171	334	202,3

PSNR	Baboon	Lena	Peppers	Sky	Média
Fidelidade	20,7	29,3	30,7	28,6	27,3
Médio	20,4	28,1	29,2	27,6	26,3
Compressão	19,1	25,8	25,8	24,9	23,9
Extremo	19,0	25,2	25,1	24,3	23,4

tabela 5.5 - Resultados práticos do algoritmos de camadas dependentes com domínios separados.

Mostram-se agora os gráficos com a relação existente entre a taxa de compressão e a fidelidade do nosso algoritmo seguindo os mesmos critérios dos apresentados no algoritmo anterior.

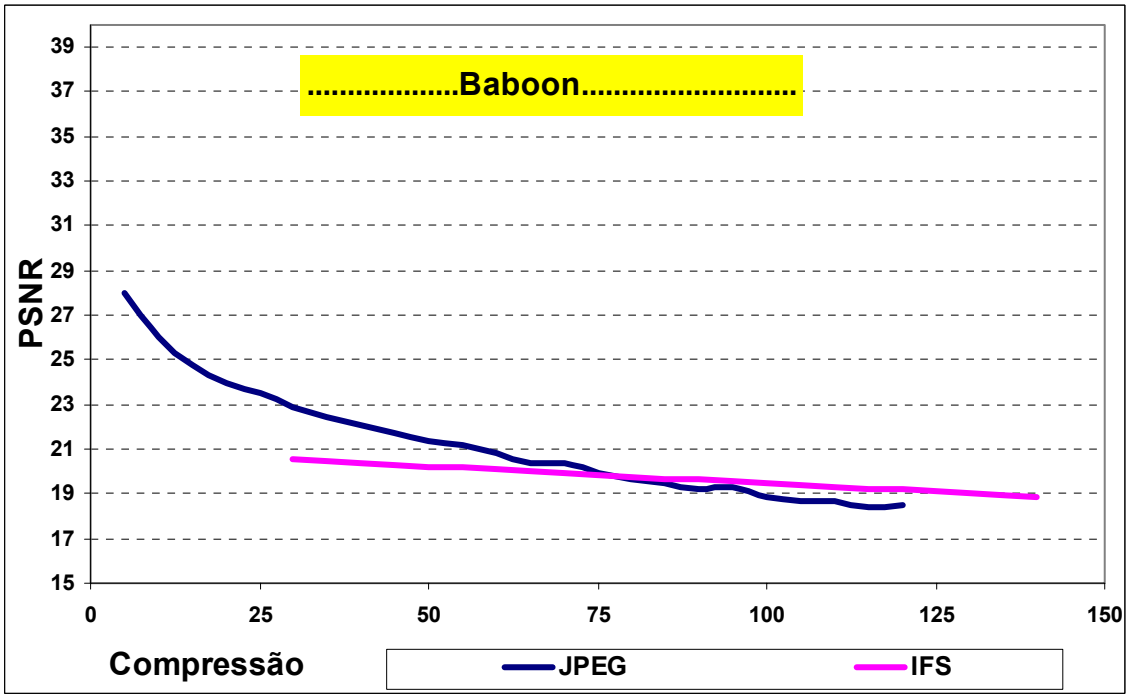


gráfico 5.9 - Resultados práticos da imagem *Baboon* com o método das camadas dependentes e domínios separados.

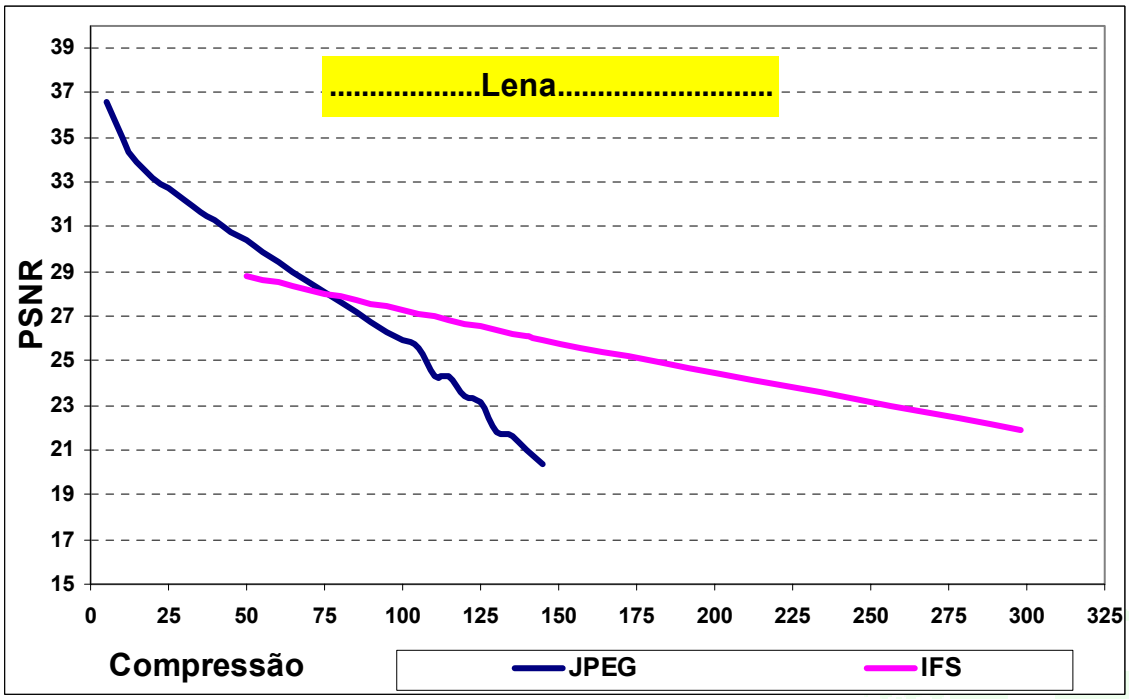


gráfico 5.10 - Resultados práticos da imagem *Lena* com o método das camadas dependentes e domínios separados.

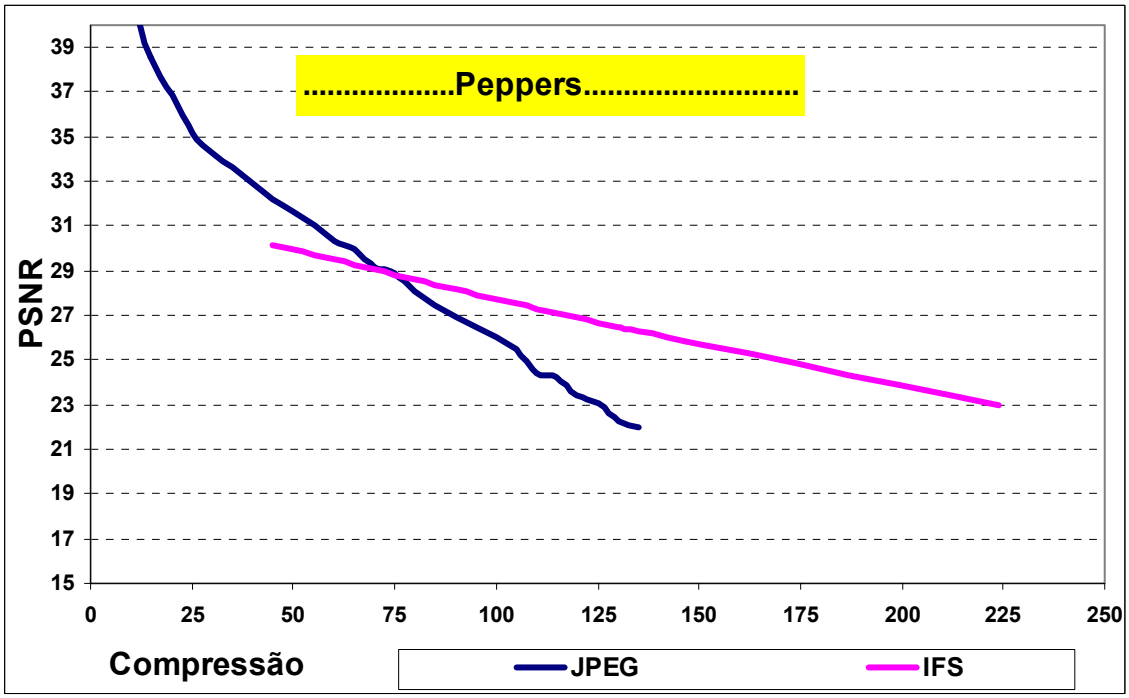


gráfico 5.11 - Resultados práticos da imagem *Peppers* com o método das camadas dependentes e domínios separados.

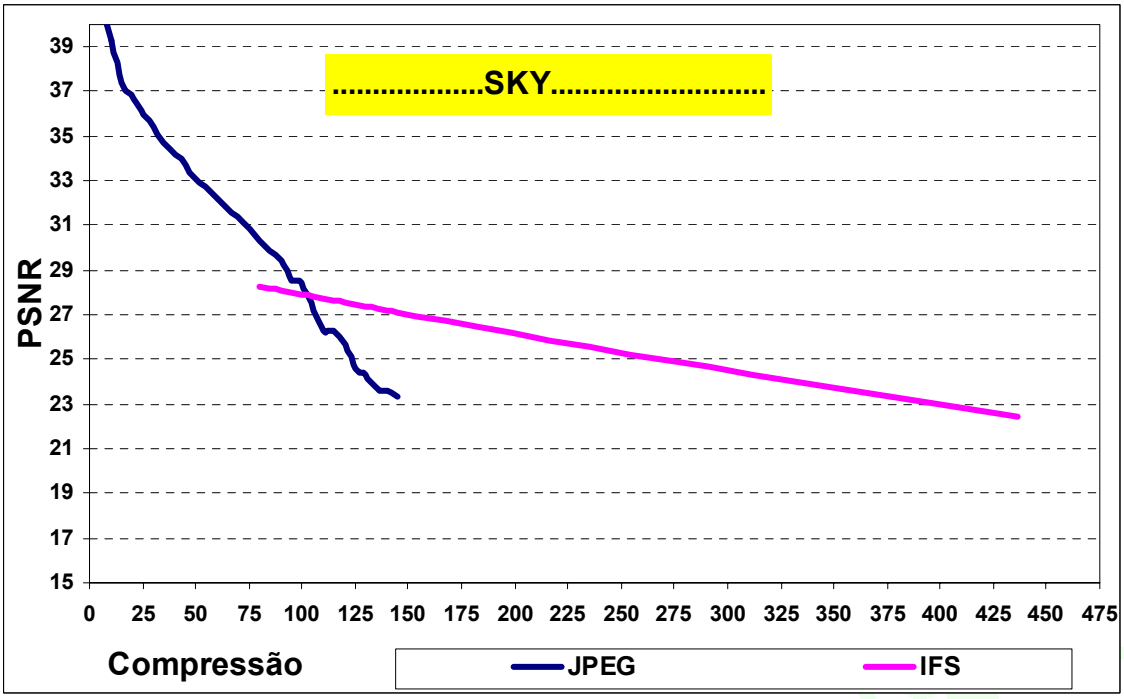


gráfico 5.12 - Resultados práticos da imagem *Sky* com o método das camadas dependentes e domínios separados.

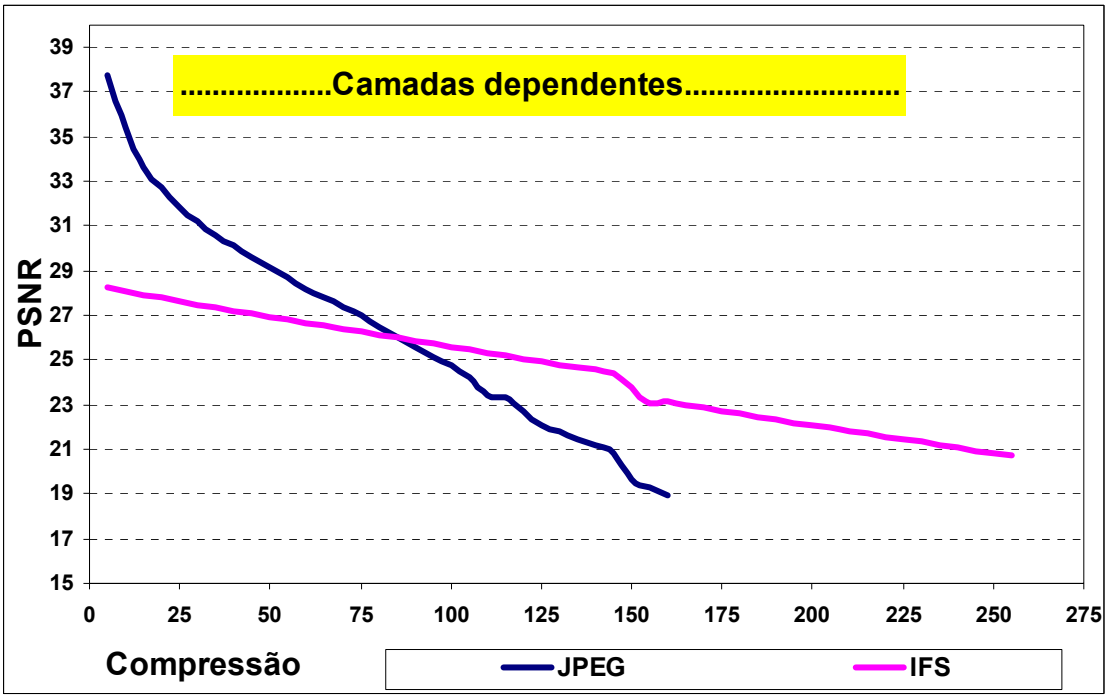


gráfico 5.13 - Média dos resultados práticos com o método das camadas dependentes e domínios separados.

Crítica dos resultados

Os resultados obtidos por este método são um pouco frustrantes, pois o algoritmo propõe um tipo de transformações que ocupam bastante menos espaços que as originais, e, no entanto, as taxas de compressão são bastante modestas. Para além disso, o método dispende muito tempo na execução da compressão.

Estes resultados explicam-se pelo facto de estarmos a tentar mapear três regiões distintas em três domínios distintos com um erro máximo admissível, o que quer dizer que se uma das transformações tiver um erro superior o bloco é dividido independentemente do erro obtido pelas outras duas transformações. Isto traduz-se numa maior divisão da imagem e, consequentemente, num aumento do número das transformações.

Como cada transformação tem um conjunto de três regiões, é necessário executar três comparações com os respectivos três domínios, antes da decisão de se dividir ou não o bloco. Estes dois factores tornam o algoritmo muito lento.

A utilização do espaço YUV não se traduz numa economia de espaço nas transformações, mas ajuda na indexação do domínio, uma vez que as camadas com menos informação obtêm erros menores nas transformações afins.

Os resultados obtidos com este algoritmo, na esmagadora maioria, são piores quando comparados com os resultados obtidos no algoritmo anterior. Nenhuma imagem

conseguiu igualar os resultados obtidos pelo método anterior, com uma melhor fidelidade.

Este fraco desempenho deve-se ao facto de estarmos a indexar camadas à mesma partição, o que se traduz num aumento do erro de compressão em relação a transformações independentes.

O que se ganha em espaço com estas transformações depressa se perde no nível de partição e no erro gerado pela indexação dos domínios

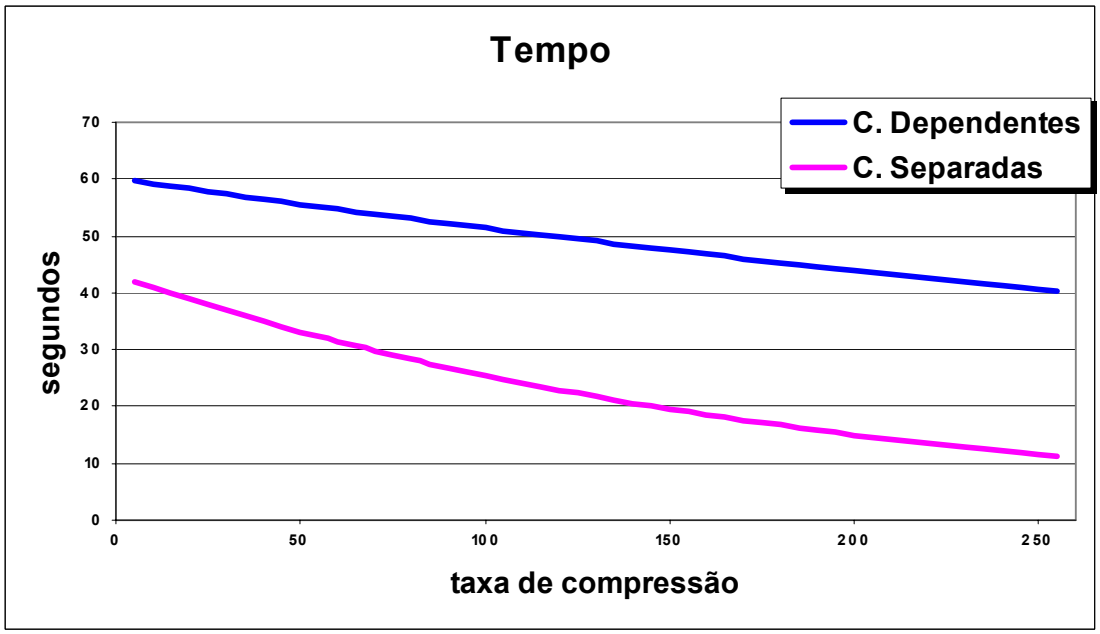


gráfico 5.14 – Tempo de compressão (segundos) do método de camadas dependentes e camadas separadas com domínios separados.

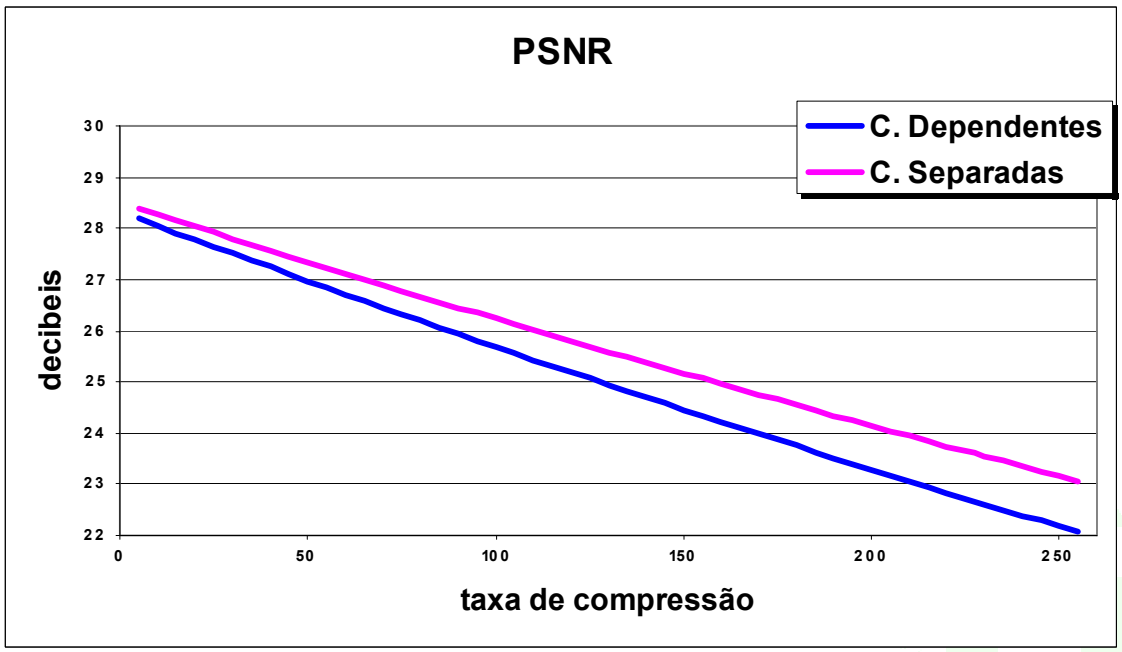


gráfico 5.15 - Fidelidade de compressão (segundos) do método de camadas dependentes e camadas separadas com domínios separados.

União dos domínios

A união dos domínios neste método vem agravar o seu desempenho em termos de tempo e taxa de compressão, e o aumento da fidelidade é insignificante. O aumento do tempo de compressão deve-se ao facto de precisar de tentar mapear a região nos domínios da cada camada. A taxa de compressão diminui, pois cada transformação tem mais seis bites do que a anterior.

Como já o algoritmo anterior é muito lento, esta versão é-o ainda mais. Sendo os resultados desanimadores, omitimos a sua apresentação neste capítulo.

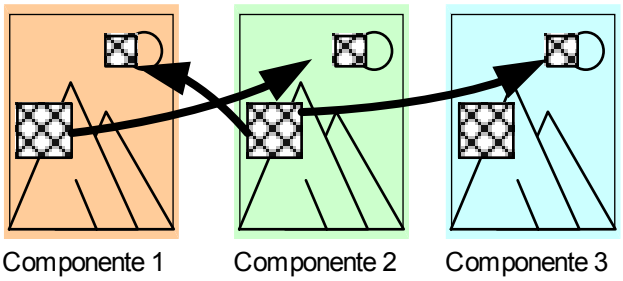


figura 5.3 – Compressão de uma região com o método de camadas dependentes e domínios unidos.

Camada principal

O terceiro método a considerar surgiu graças à existência de uma camada na imagem contendo mais informação que as outras. A camada com mais informação é definida como principal, e as restantes, como secundárias e vão ser submetidas à partição definida na camada principal.

Um dos problemas do método anterior é a pesquisa de um endereço de domínio que satisfaça simultaneamente as transformações de cada uma das camadas. O outro problema é o tempo necessário para encontrar esse mesmo endereço.

Para resolver estes problemas vamos tirar partido do espaço de cor escolhido e procurar o domínio apenas na camada com mais informação (Y). Devido à semelhança entre as camadas e uma vez que as camadas U e V contêm menos informação, assumimos que o domínio escolhido codifica correctamente as regiões destas camadas.

Em virtude da grande correlação existente entre as camadas, vamos codificá-las a todas com os domínios da camada principal.

Resultados práticos demonstram que a utilização da camada principal para a codificação das restantes aumenta a fidelidade da compressão, o que se compreende pelo facto daquela camada conter mais informação.

Esta abordagem resolve simultaneamente os dois problemas focados atrás. As regiões das camadas U e V não sofrem qualquer tipo de pesquisa, pois são

automaticamente indexadas à camada Y, o que resolve concomitantemente a questão do tempo.

As transformações afim usadas por este método são iguais às do método anterior.

A seguir são apresentados os resultados práticos da compressão com domínios separados:

Tempo	Baboon	Lena	Peppers	Sky	Média
Fidelidade	32	27	29	21	27,3
Médio	9	7	8	6	7,5
Compressão	12	9	9	5	8,8
Extremo	11	7	6	4	7,0

Compressão	Baboon	Lena	Peppers	Sky	Média
Fidelidade	29	46	45	84	51,0
Médio	32	58	61	107	64,5
Compressão	130	149	142	253	168,5
Extremo	135	179	174	339	206,8

PSNR	Baboon	Lena	Peppers	Sky	Média
Fidelidade	20,9	29,8	30,9	29,0	27,7
Médio	20,6	28,8	29,5	28,0	26,7
Compressão	19,2	26,0	26,0	25,2	24,1
Extremo	19,2	25,4	25,4	24,5	23,6

tabela 5.6 - Resultados práticos do algoritmos de camada principal com domínios separados.

Apresentamos de seguida os gráficos com a relação existente entre a taxa de compressão e a fidelidade do método de camada principal:

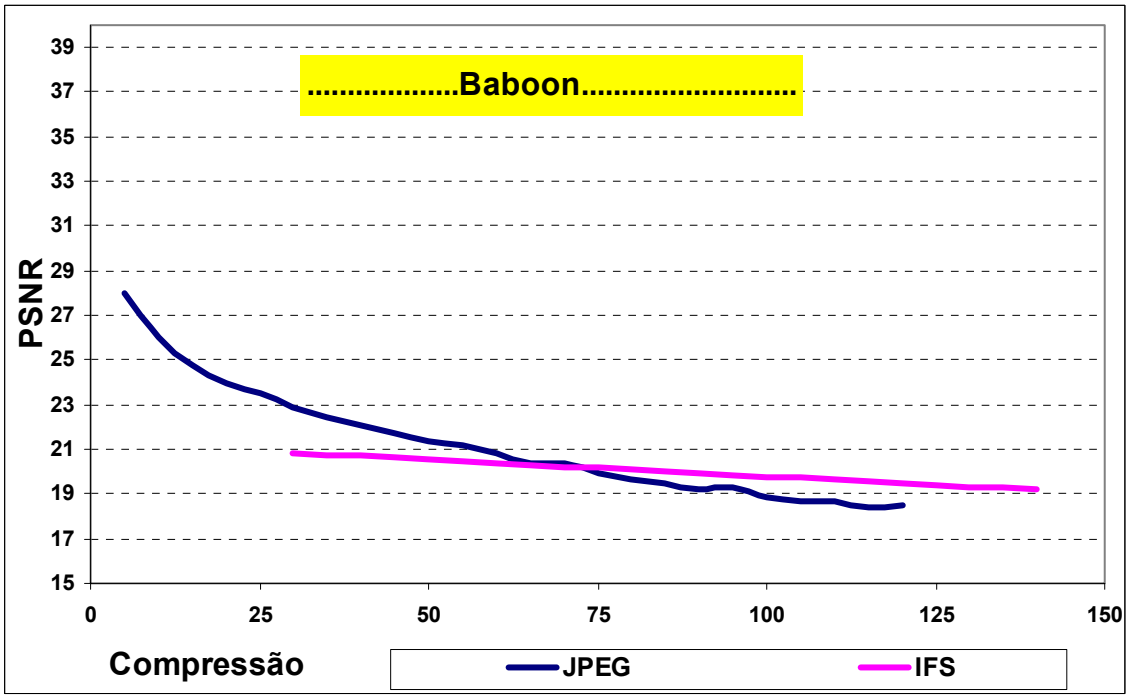


gráfico 5.16 - Resultados práticos da imagem *Babbon* com o método da camada principal e domínios separados.

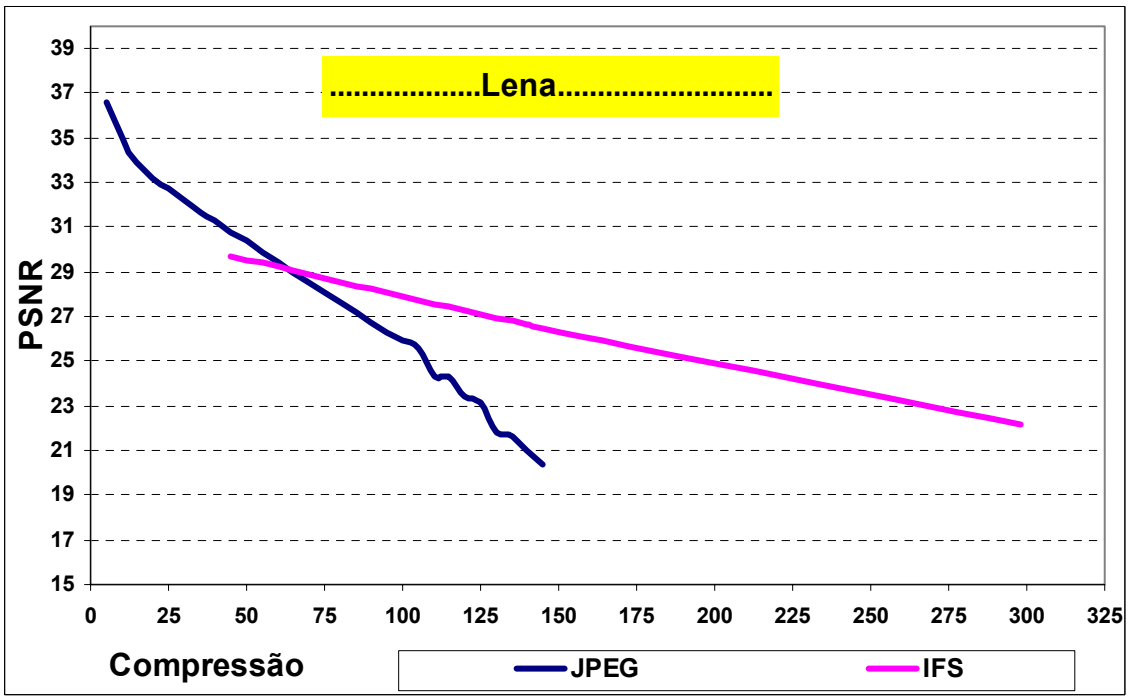


gráfico 5.17 - Resultados práticos da imagem *Lena* com o método da camada principal e domínios separados.

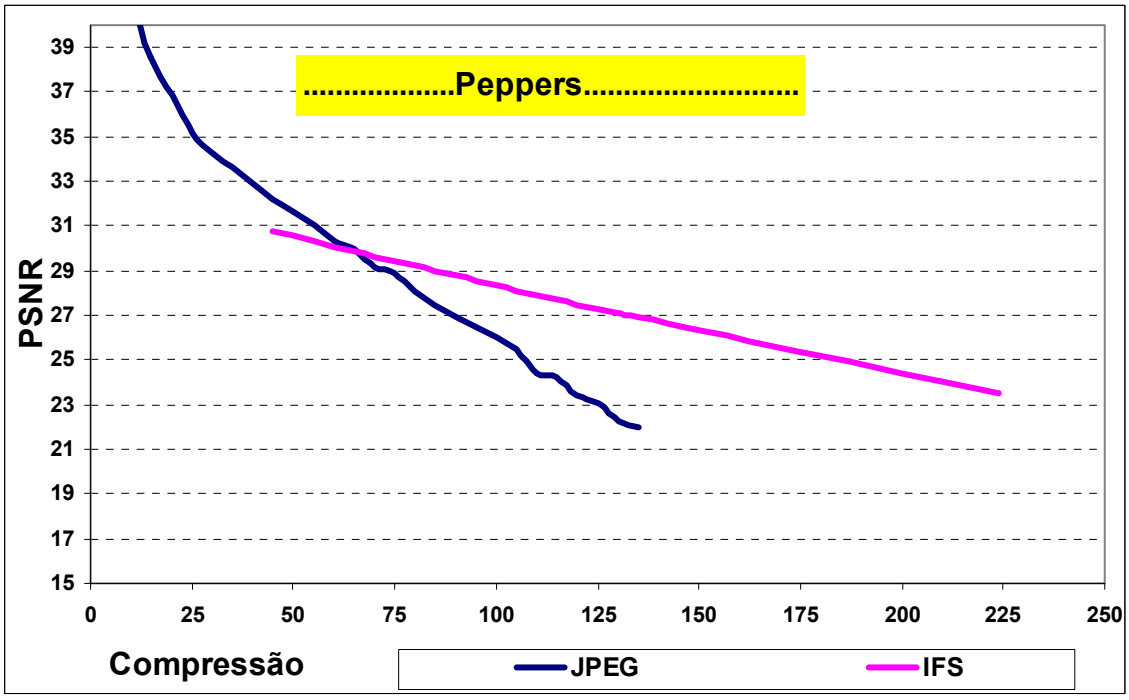


gráfico 5.18 - Resultados práticos da imagem *Pepers* com o método da camada principal e domínios separados.

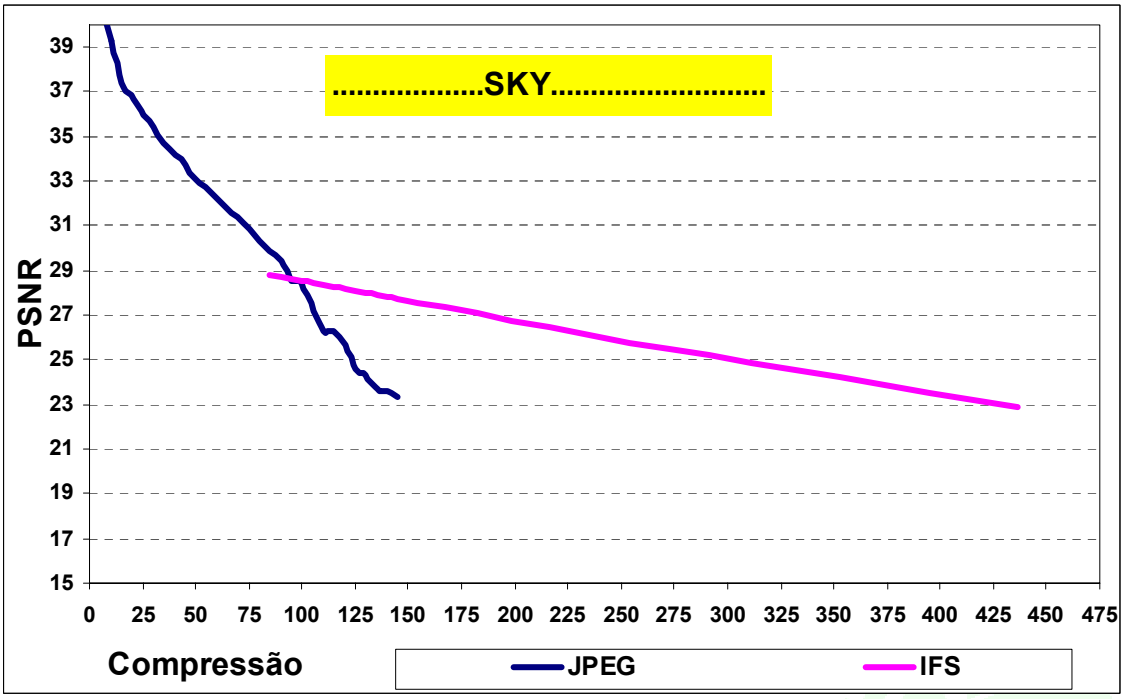


gráfico 5.19 - Resultados práticos da imagem *Sky* com o método da camada principal e domínios separados.

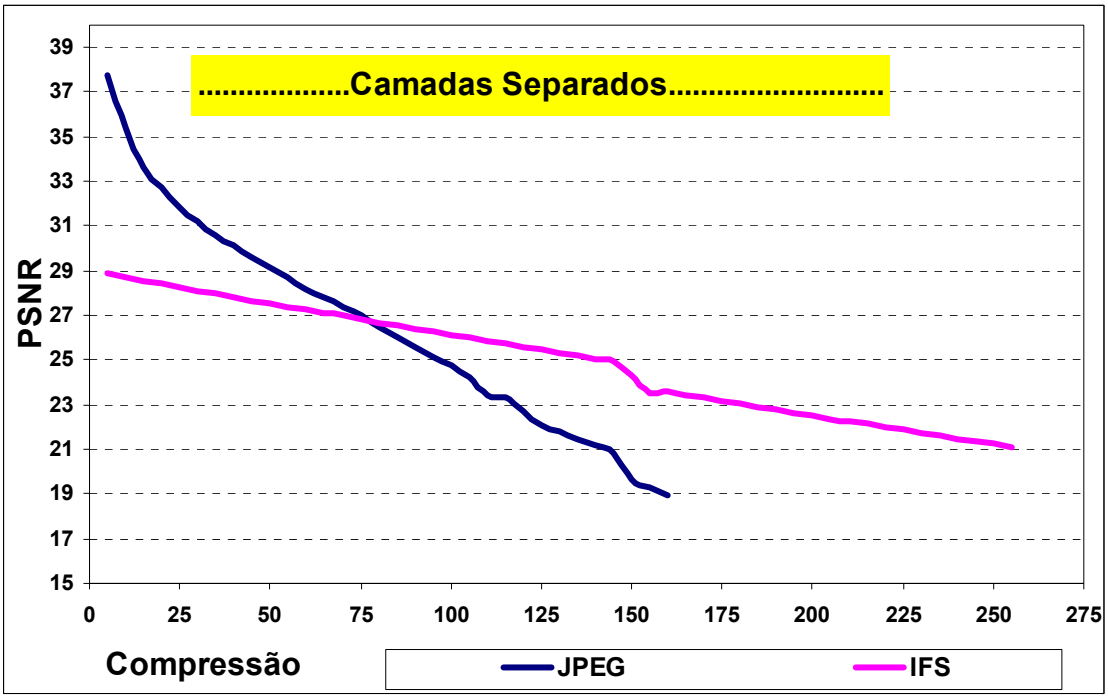


gráfico 5.20 - Média dos resultados práticos com o método da camada principal e domínios separados.

Crítica aos resultados

O tempo de compressão sofreu uma redução significativa, pois a pesquisa do domínio é efectuada apenas numa camada, enquanto que as restantes aproveitam as coordenadas da camada principal para calcularem o seu factor de escala e de deslocamento.

A taxa de compressão situa-se ao mesmo nível do algoritmo de camadas separadas. No entanto, os resultados são obtidos muito mais rapidamente.

A fidelidade é comparável à obtida pelos restantes métodos, o que demonstra que podemos indexar as camadas secundárias à camada principal, sem grande perda da qualidade de compressão.

Mostramos de seguida a comparação gráfica deste método com o método das camadas separadas em termos de tempo e de fidelidade:

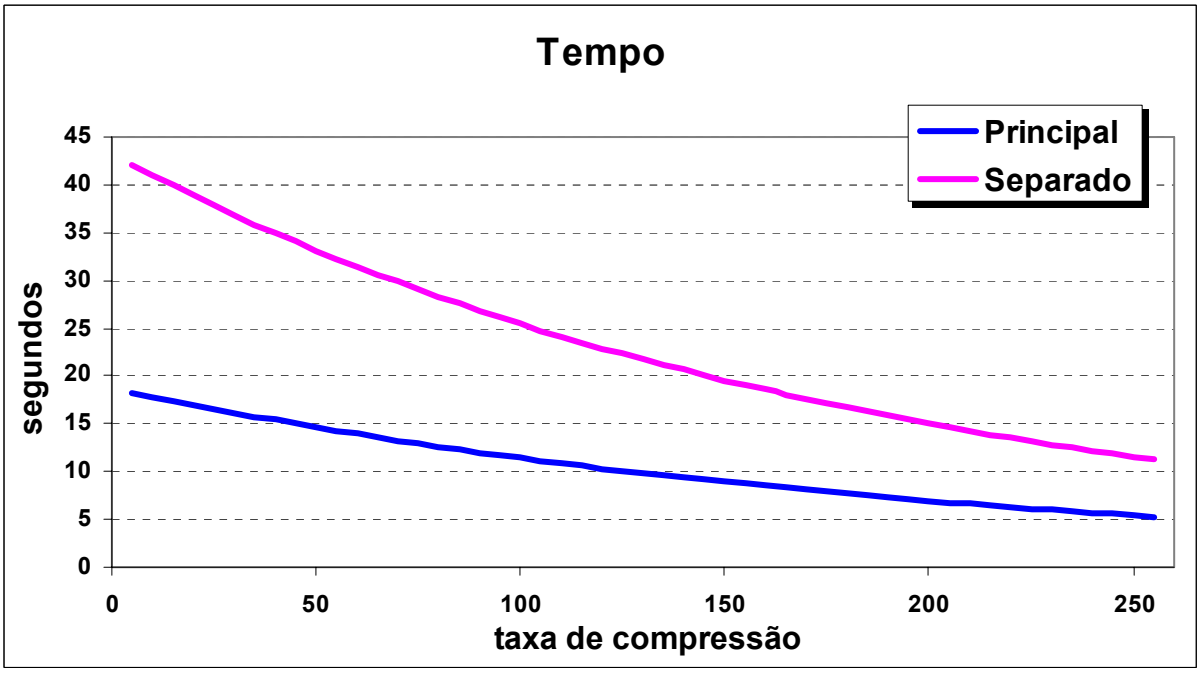


gráfico 5.21 - Tempo de compressão (segundos) do método de camada principal e de camadas separadas com domínios separados.

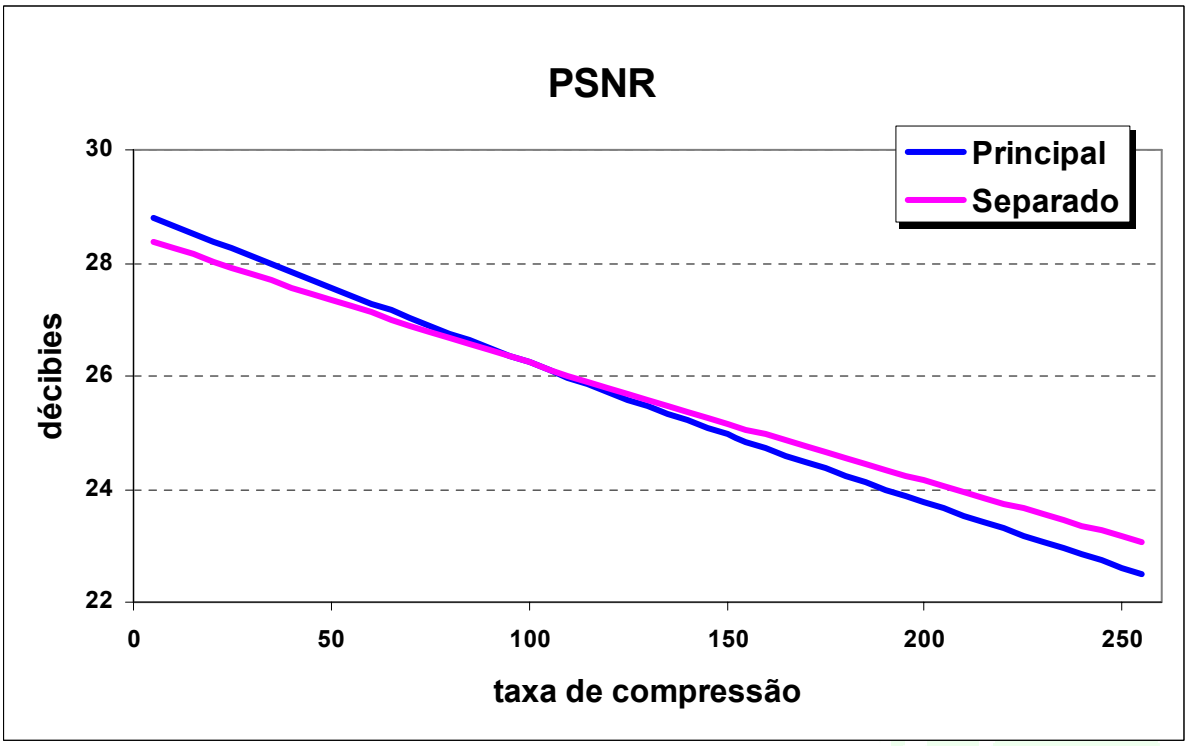


gráfico 5.22 - Fidelidade da compressão (db) do método de camada principal e de camadas separadas com domínios separados.

União dos domínios

No algoritmo de camadas separadas a união dos domínios deu resultados bastante fracos, e, após várias versões para essa mesma união, chegamos à conclusão que a qualidade obtida era diminuta e que o tempo de compressão aumenta para o triplo. Como a grande vantagem deste algoritmo em relação aos restantes é a velocidade de compressão, vamos omitir deste trabalho a apresentação destes resultados.

Camadas reduzidas

Este método, tal como o anterior, visa tirar partido da existência de camadas com menos informação. Por isso, este método aplica-se a imagens nos espaços de cor YUV e similares. Também é possível comprimir imagens noutros formatos de cor, mas a fidelidade é muito menor.

Como as camadas com menos informação têm uma participação menor na reconstrução da imagem original no espaço RGB, aquelas podem ser codificadas de uma forma mais grosseira.

As camadas com menos informação são reduzidas com um factor de dois na altura e na largura. As camadas assim obtidas são codificadas com o método de camadas separadas. A taxa de compressão conseguida por cada uma das camadas reduzidas tem um efeito quatro vezes superior na imagem principal.

A falta de dimensão das transformações afim possibilita a descompressão de uma imagem para qualquer dimensão. Por esta razão, este método tira partido desta característica de forma a descomprimir as camadas reduzidas para o seu tamanho original.

Os resultados práticos da compressão com domínios separados são visíveis na tabela abaixo exposta:

Tempo	**Baboon**	**Lena**	**Peppers**	**Sky**	**Média**
Fidelidade	33	29	31	22	28,8
Médio	11	8	9	7	8,8
Compressão	15	11	12	6	11,0
Extremo	15	7	8	4	8,5

Compressão	**Baboon**	**Lena**	**Peppers**	**Sky**	**Média**
Fidelidade	39	70	61	120	72,5
Médio	48	95	88	164	98,8

Compressão	Baboon	Lena	Peppers	Sky	Média
Compressão	187	220	203	397	251,8
Extremo	221	324	274	570	347,3

PSNR	Baboon	Lena	Peppers	Sky	Média
Fidelidade	20,8	29,0	30,1	28,6	27,1
Médio	20,4	27,3	27,9	27,3	25,7
Compressão	19,1	25,5	25,2	24,6	23,6
Extremo	18,9	23,7	23,8	23,7	22,5

tabela 5.7 - Resultados práticos do algoritmo de camadas reduzidas.

Apresentamos agora os gráficos com a relação existente entre a taxa de compressão e a fidelidade do método das camadas reduzidas:

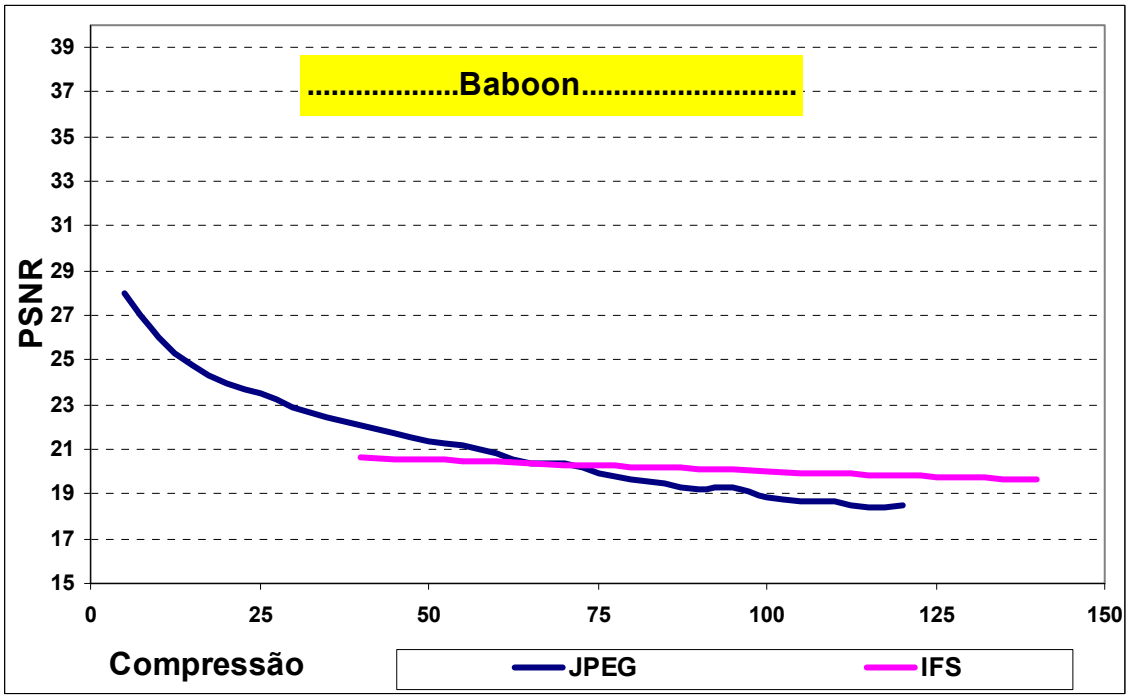


gráfico 5.23 - Resultados práticos da imagem *Babbon* com o método de camadas reduzidas.

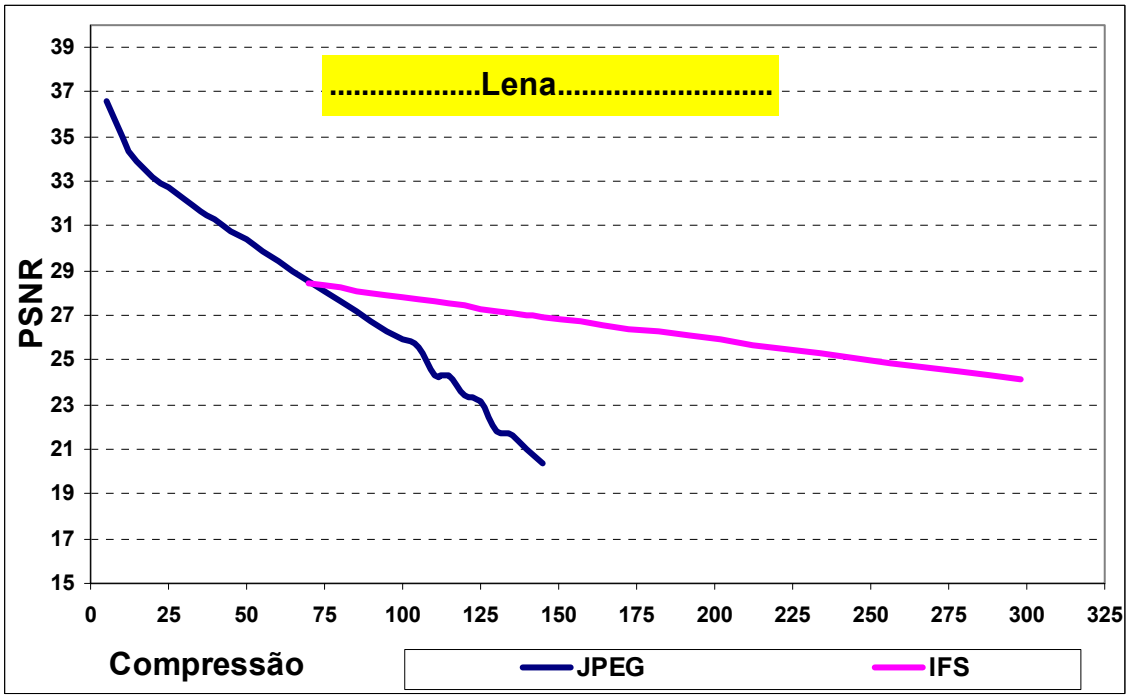


gráfico 5.24 - Resultados práticos da imagem *Lena* com o método de camadas reduzidas.

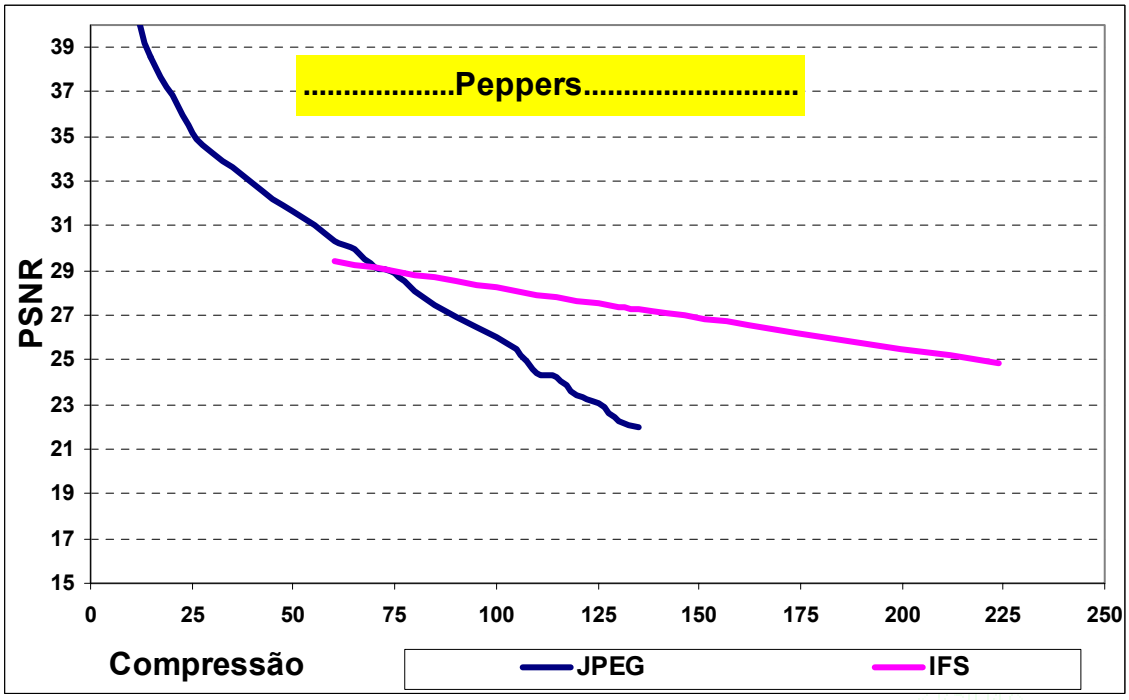


gráfico 5.25 - Resultados práticos da imagem *Peppers* com o método de camadas reduzidas.

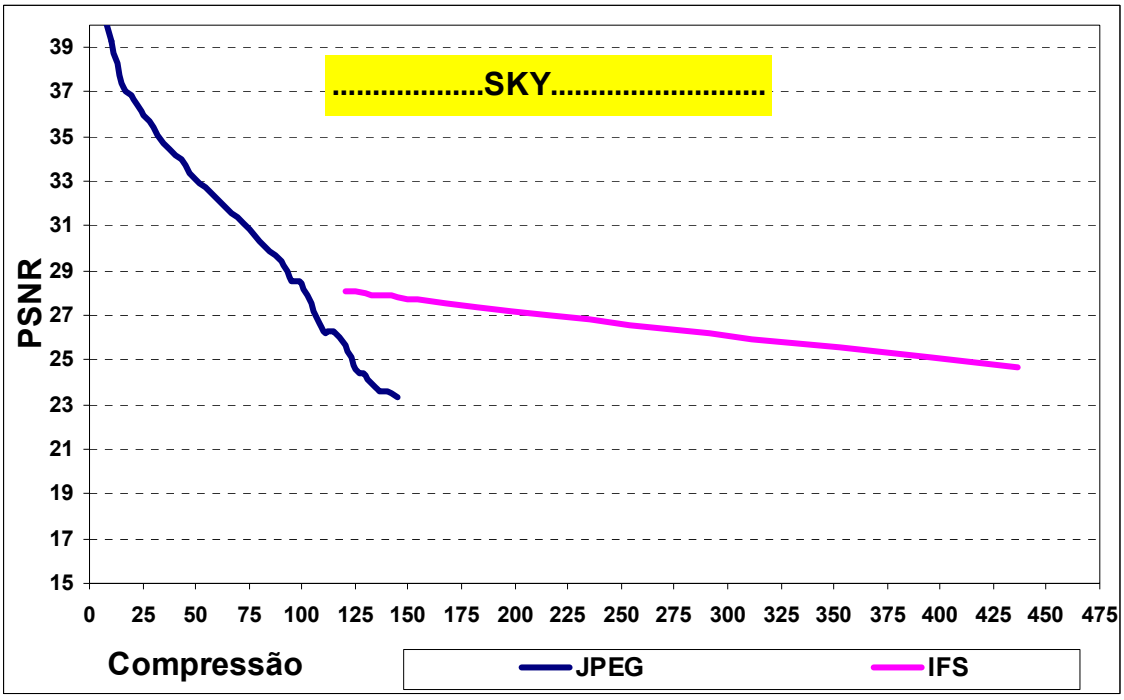


gráfico 5.26 - Resultados práticos da imagem *Sky* com o método de camadas reduzidas.

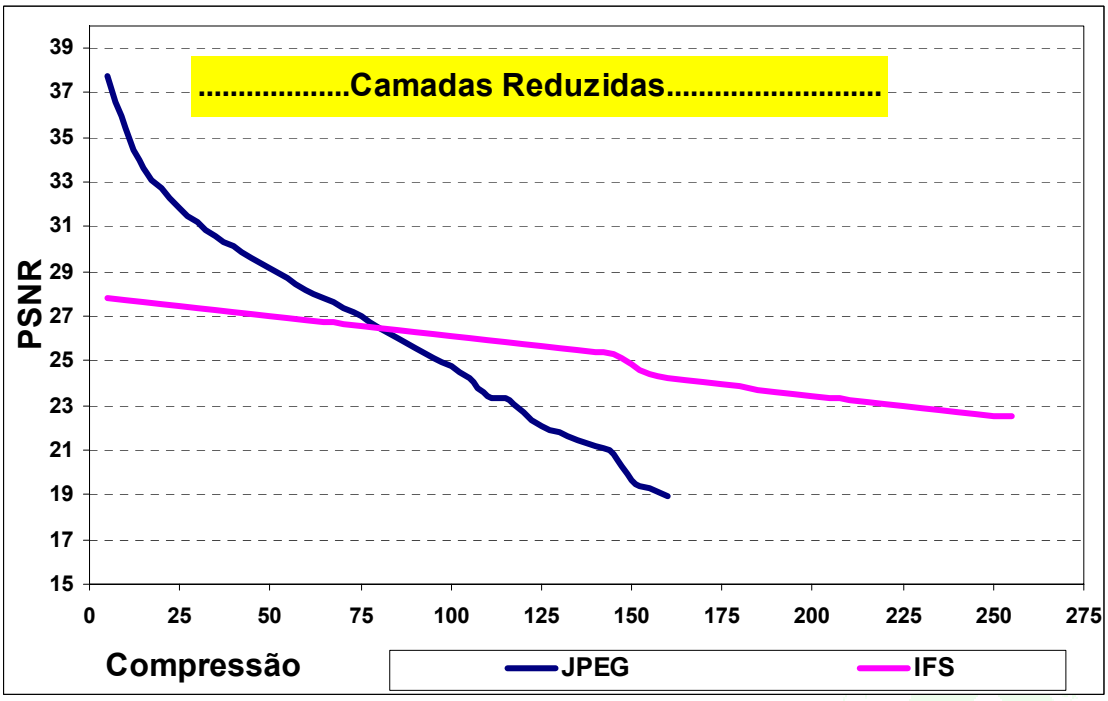


gráfico 5.27 - Média dos resultados práticos com o método de camadas reduzidas.

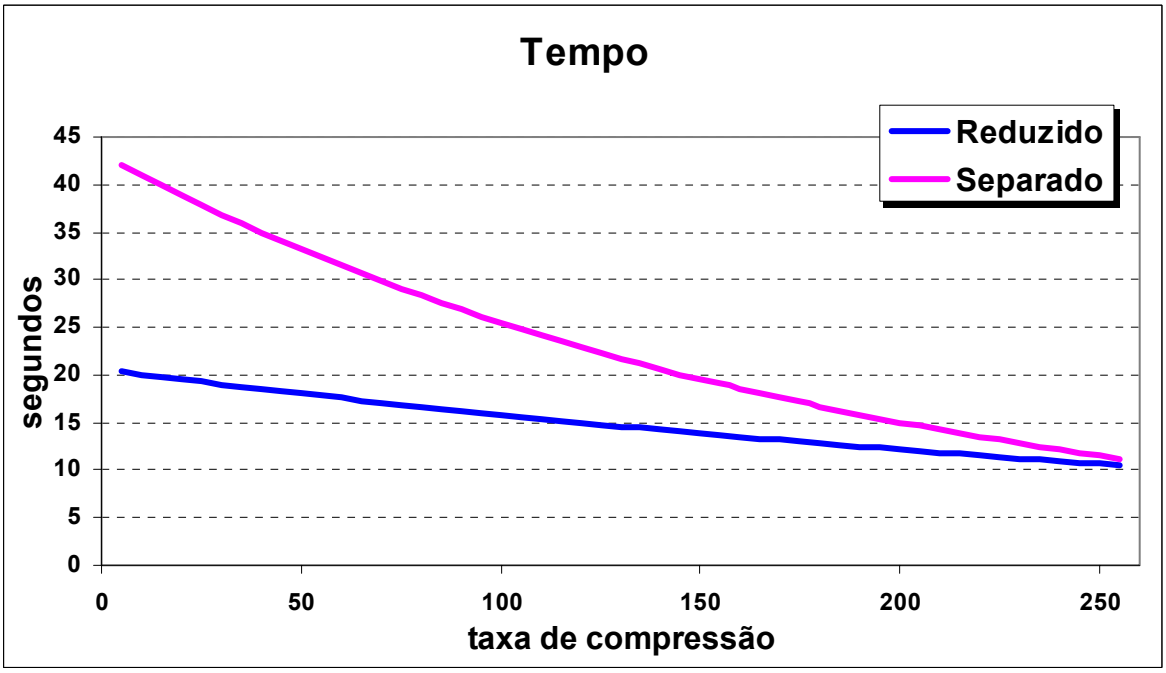


gráfico 5.28 – Comparação da média do tempo de compressão entre o método de camadas reduzidas e o de camadas separadas.

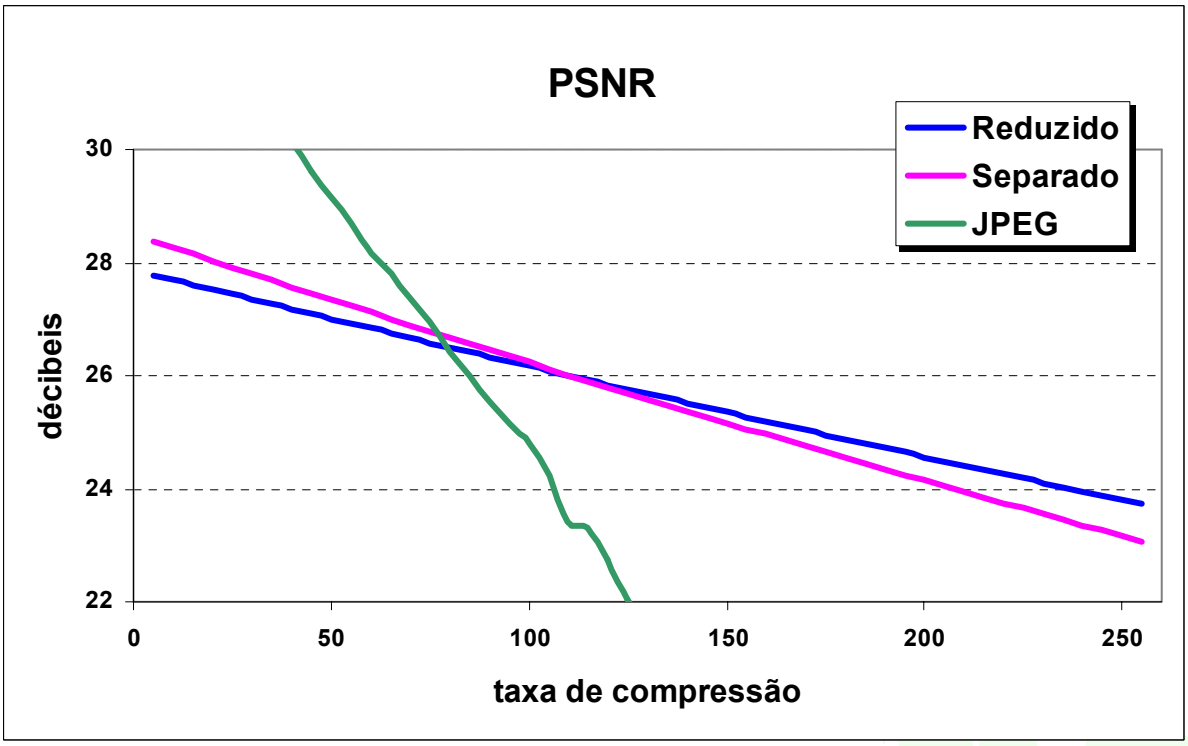


gráfico 5.29 - Comparação da fidelidade de compressão entre o método de camadas reduzidas e o de camadas separadas JPEG.

Crítica aos resultados

Se comparamos os resultados deste algoritmo com o de camadas separadas concluímos que a taxa de compressão aumentou de forma significativa em todos os modos pré-definidos. Obviamente, este aumento é acompanhado pela perda de fidelidade; contudo, o saldo é positivo para taxas de compressão altas, como se pode ver nos gráficos anteriores.

O tempo necessário para a compressão baixou igualmente, pois existem duas camadas quatro vezes mais pequenas.

Algoritmo híbrido

Esta solução é uma variação do algoritmo anterior. Ao contrário deste, as duas camadas secundárias são codificadas com o método de *camada principal*, ou seja, uma das camadas codificadas é submetida à codificação da outra.

Este algoritmo visa o aumento da taxa de compressão e a diminuição do tempo necessário para a conclusão da mesma.

A seguir são apresentados os resultados práticos:

Tempo	**Baboon**	**Lena**	**Peppers**	**Sky**	**Média**
Fidelidade	27	28	28	20	25,8
Médio	8	7	7	6	7,0
Compressão	13	9	10	6	9,5
Extremo	11	7	7	4	7,3

Compressão	**Baboon**	**Lena**	**Peppers**	**Sky**	**Média**
Fidelidade	43	71	63	126	75,7
Médio	52	98	88	168	101,5
Compressão	203	234	210	413	265,0
Extremo	233	333	263	585	353,5

PSNR	**Baboon**	**Lena**	**Peppers**	**Sky**	**Média**
Fidelidade	20,8	27,8	29,1	28,6	26,6

PSNR	Baboon	Lena	Peppers	Sky	Média
Médio	20,3	26,4	27,5	27,3	25,4
Compressão	19,1	24,9	24,9	24,6	23,4
Extremo	18,8	23,2	23,8	23,7	22,4

tabela 5.8 - Resultados práticos do algoritmo híbrido.

Os gráficos da relação taxa de compressão / fidelidade são muito semelhantes aos do método anterior, porque o aumento da taxa de compressão é acompanhado por um decréscimo na qualidade de compressão.

Expomos a seguir os gráficos com a comparação do tempo de compressão entre os algoritmos de camadas reduzidas e o algoritmo híbrido, bem como a comparação da relação compressão / fidelidade dos dois algoritmos anteriores e o algoritmo JPEG.

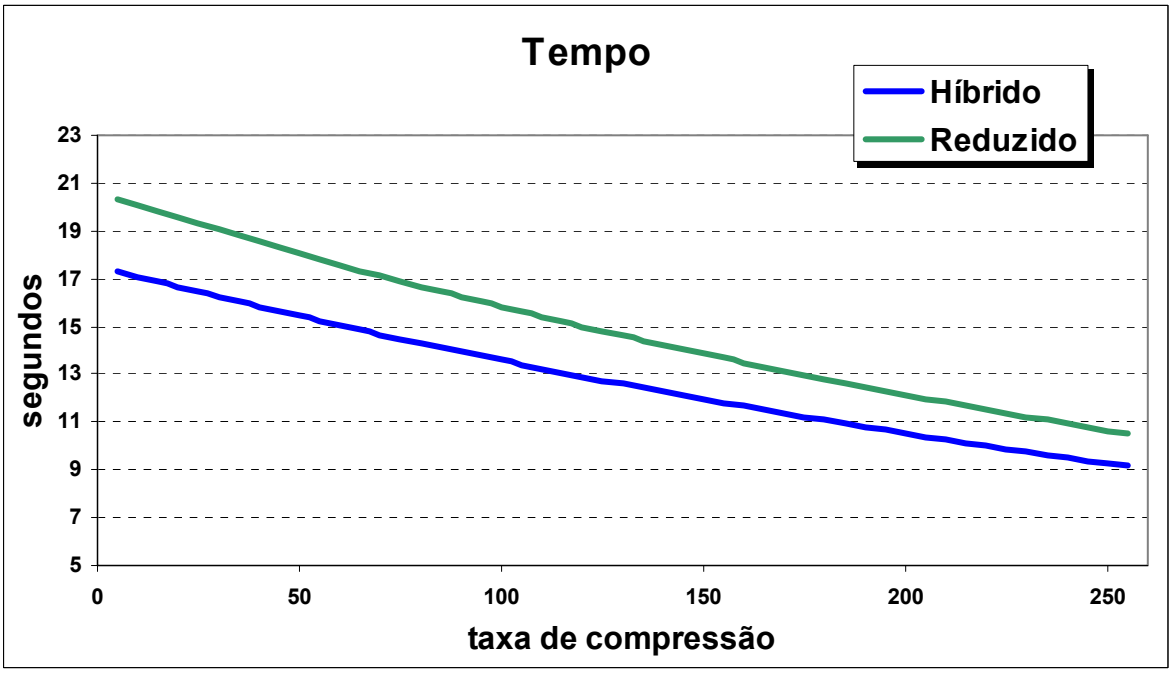


gráfico 5.30 – Tempo de compressão do método híbrido e do método de camadas reduzidas.

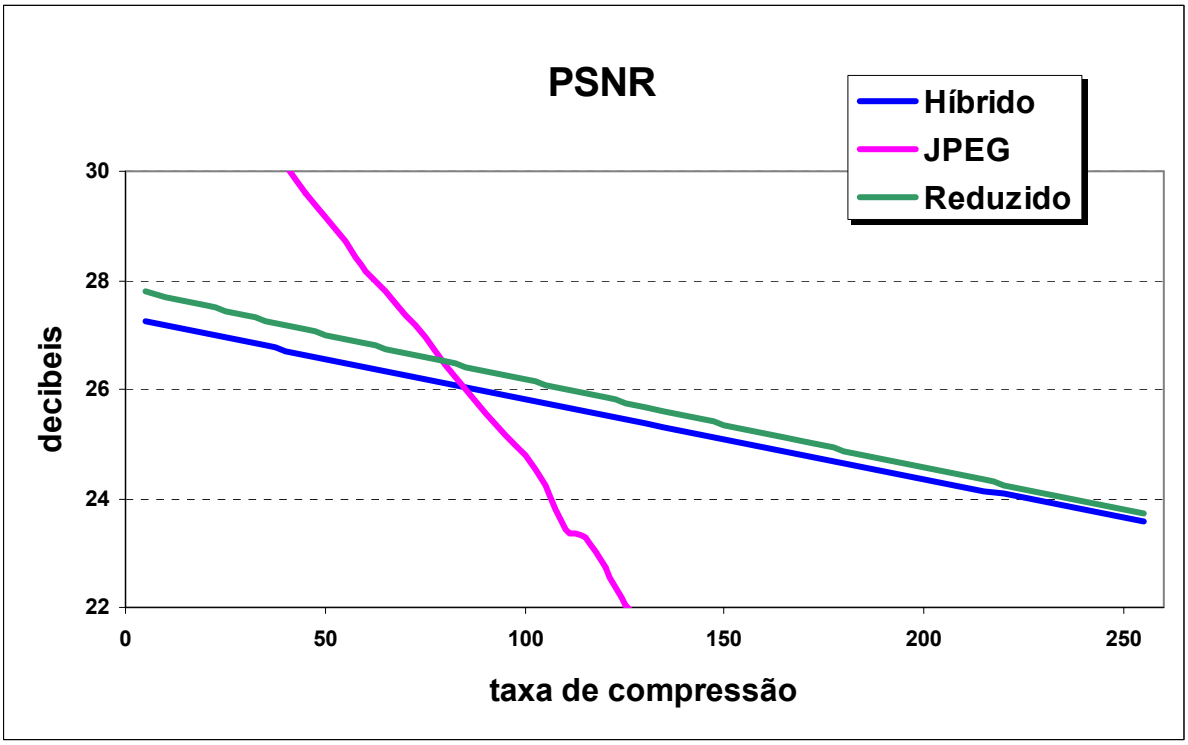


gráfico 5.31 – Relação compressão/fidelidade do método híbrido do método de camadas reduzidas e do algoritmo JPEG.

Crítica aos resultados

O algoritmo híbrido correspondeu de forma positiva aos objectivos traçados. Este, na realidade, comprime mais e mais rápido do que o algoritmo de camadas reduzidas. Todavia, a fidelidade de compressão foi afectada de forma negativa.

No cômputo geral, concluímos que o ganho com a submissão de uma camada secundária à outra na compressão não compensa a perda de fidelidade.

Os ganhos em termos de tempo são indiscutivelmente positivos, mas, tomando em consideração que o algoritmo anterior já tinha um bom desempenho, estes ganhos são atenuados.

Descompressão das imagens

A expansão das imagens comprimidas passa por aplicar cada uma das transformações afins a uma imagem qualquer, construindo desta forma uma nova imagem. O processo repete-se aplicando o PIFS à imagem obtida na iteração anterior até se obter o ponto fixo, que representa a imagem descomprimida. Para este processo são necessárias duas imagens: a imagem original e a imagem obtida pelo PIFS.

O processo anterior converge para a imagem com cerca de uma dezena de iterações, no entanto o processo pode ser acelerado se utilizarmos apenas uma imagem.

Isto permite que as transformações afins utilizem a imagem já descomprimida pelas transformações anteriores.

Esta nova versão reduz o número de iterações, mas também, converge muito mais depressa.

A tabela seguinte mostra as primeiras quatro iterações da imagem Lena através do novo método de descompressão.

tabela 5.9 – As quatro primeiras iterações da descompressão da imagem Lena pelo método de descompressão modificado.

O gráfico seguinte apresenta os resultados da descompressão de imã imagem pelo método fornecido por Yuval Fisher e os resultados do método modificado.

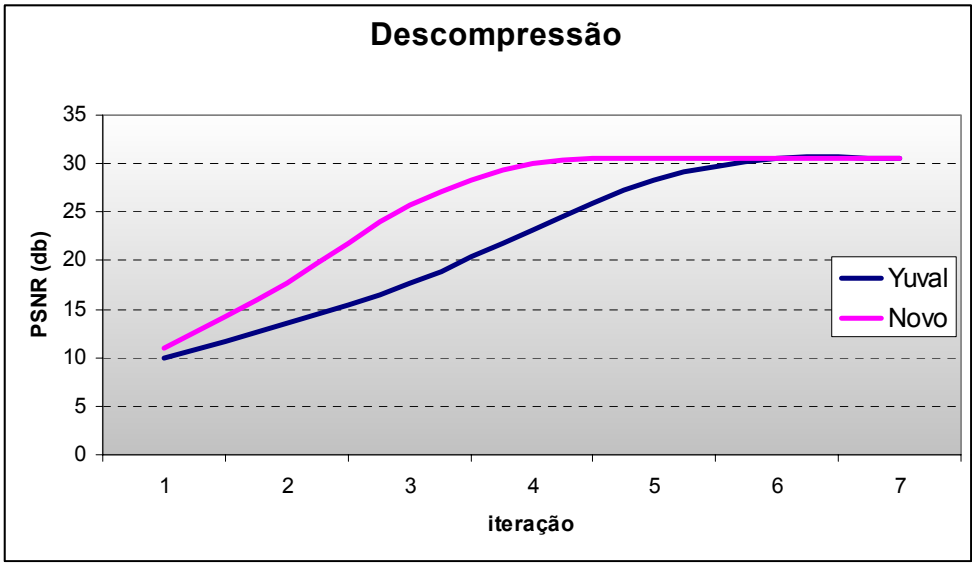


gráfico 5.32 – Descompressão da imagem Lena pelo método tradicional e pelo método modificado.

Conclusão

Dos resultados práticos dos algoritmos aqui apresentados, podemos tecer as seguintes considerações:

- os nossos algoritmos têm uma relação assimétrica entre a taxa de compressão e o respectivo tempo. Quanto maior for a taxa de compressão, menor é o tempo de

execução do algoritmo. De acordo com os parâmetros definidos para a compressão, uma grande taxa de compressão implica a existência de poucas transformações, o que permite reduzir o número de pesquisas, economizando, assim, tempo de compressão.

- O algoritmo JPEG obtém excelentes resultados para taxas de compressão baixas. Porém, a fidelidade de compressão com taxas próximas da centena degrada-se bastante, e é aqui que os nossos algoritmos começam a ter um interesse prático real.
- Em termos de taxa de compressão, o algoritmo de camadas reduzidas é o que comprime mais com a melhor relação qualidade / quantidade.
- O algoritmo de camada principal é o que melhor desempenho tem em relação ao tempo de compressão.
- O algoritmo de camadas separadas perde o seu interesse, no que concerne à taxa de compressão, em favor do algoritmo de camadas reduzidas e em relação ao tempo para os dois.

A seguir é apresentado um resumo dos métodos de compressão implementados sobre a forma de gráficos.

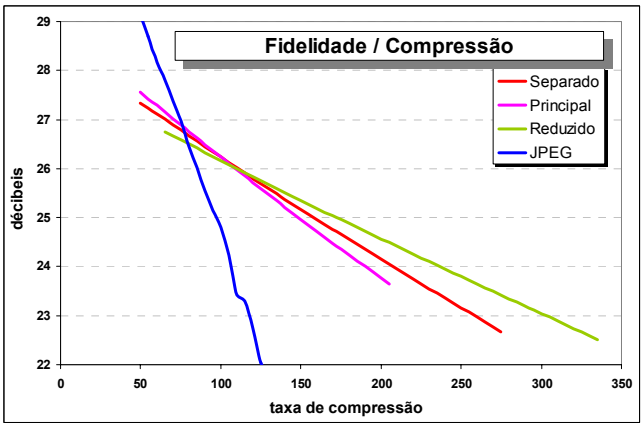


gráfico 5.33 - Comparação da taxa de compressão dos algoritmos IFS com o algoritmo JPEG

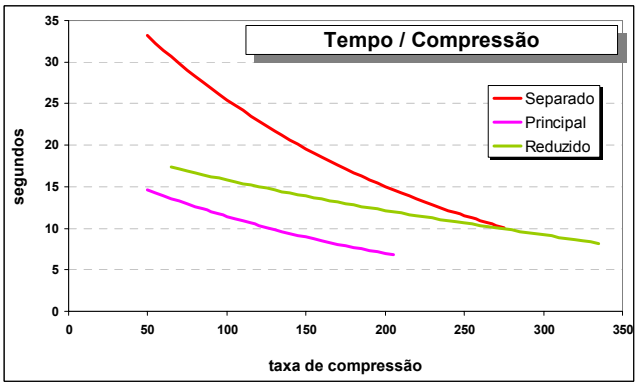


gráfico 5.34 - Comparação do tempo de compressão dos algoritmos IFS.

Capítulo 6: Conclusão

Os algoritmos implementados funcionam bem e têm resultados promissores. Tentamos com este trabalho explorar uma faceta desta tecnologia que tem sido tratada de uma forma muito ligeira nas publicações a que tivemos acesso.

Nos testes realizados, imprimimos sem um cunho de aplicabilidade real muito forte. A esmagadora maioria das publicações que serviram de suporte a este trabalho apresentam esta tecnologia como capaz de executar compressões de qualidade semelhante ou até superior ao JPEG. Embora isso seja de louvar, e muitos melhoramentos daí resultantes estão agora presentes na tecnologia, o tempo que leva a conseguir os resultados é assustador, relegando a tecnologia para compressões em diferido.

Neste trabalho, seguimos uma linha um pouco diferente: tornar as compressões rápidas e comprimir além dos limites da tecnologia JPEG. As compressões não têm uma qualidade fantástica, mas conseguimos obter imagens com poucos kilobytes, com compressões na ordem das várias centenas.

Este tipo de compressões tem interesse acrescido quando a largura de banda é escassa, ou dispendiosa, e quando o detalhe das imagens pode ser relegado para segundo plano.

Um exemplo bem ilustrativo da boa aplicabilidade desta técnica é o sector das imagens meteorológicas, onde o detalhe das imagens não é significativo: por exemplo, a identificação de formações nebulosas não precisa, em geral, de grandes detalhes.

Trabalho futuro

Neste trabalho não se procedeu à compressão dos coeficientes que compõem as transformações afim, de forma que, técnicas de predição e de codificação estatísticas podem possibilitar o aumento da taxa de compressão.

Com a utilização de espaços de cor cuja densidade de informação em cada camada é significativa, a codificação dos coeficientes com um número distintos de bites para cada camada é uma possibilidade real e explorada por outros algoritmos de compressão.

Como esta tecnologia está ainda em estudo e desenvolvimento, dá a este trabalho um interesse possivelmente efémero, no entanto, o que pode constituir uma contribuição mais duradoura talvez seja a opção por uma configuração de compressão fractal para taxas de compressão elevadas, onde os resultados obtidos são animadores, e os algoritmos concorrentes não abundam.

Novos algoritmos estarão, certamente, a serem forjados nos laboratórios dos cientistas que se debruçam sobre esta áreas. Essas melhorias poderão, com certeza, ser aplicadas aos métodos que aqui exploramos.

Bibliografia

[ALM97] AU, O.C.; LIOU, M.L.; MA, L.K - *Fast fractal encoding in frequency domain* - Santa Barbara, California - Proc. ICIP-97 IEEE International Conference on Image Processing – 1997.

[ANS97] ASGARI, Saeed ; NGUYEN, T.Q.; SETHARES, William A. - *Wavelet-based fractal transforms for image coding with no search* - Santa Barbara, California - Proc. ICIP-97 IEEE International Conference on Image Processing – 1997.

[APO98] *Apostila de Computação gráfica* – Rio de Janeiro - CCE/PUC – 1998.

[ARE95] AREAL, Zita – *Visualmente* – Areal Editores – 1995.

[BAR88] BARNSLAY, Michael - *Fractals Everywhere* - Academic Press, 1988.

[BAR96] BARNSLEY, Michael F. – *Fractal Image Compression – in Notices of the AMS* – Volume 43, nº 6 – 1996.

[BEE01] BEEGAN, Andrew P. – *Wavelet-Based Image Compression Using Human Visual System Models* – Virginia - Virginia Polytechnic Institute and State University – 2001.

[BRAN99] BRANDALIZE, Amauri Alfredo – *Compressão em arquivos digitais de imagens: onde estamos e para onde vamos* - Esteio Engenharia e Aerolevantamentos S.A., Paraná, Brasil - 1999

[BRE99] BREAZU, M. - *Rate-distorsion optimized quadtree partitioning method* - Delft - Proc. of the Conference Fractals in Engineering, pp. 36-42 - 1999.

[BVO94] BARTHEL, K. U. ; VOYÉ, T. - *Adaptive fractal image coding in the frequency domain* - Budapest - Proceedings of International Workshop on Image Processing: Theory, Methodology, Systems and Applications - 1994.

[CAR01] CARDINAL, Jean – "Fast Fractal Compression of Greyscale Images" – *in IEEE Transactions on Image Processing*, Vol 10, nº 1 – 2001.

[CDB93] CHASSERY, J.M.; DAVOINE, Frank; BERTIN, E. - *Compression fractale par partitionnement de delaunay* - Juan-les-Pins - In Proc. of 14th Conference GRETSI, Vol. 2, pp. 819-822 - 1993.

[DAV95] DAVOINE, Franck – *Compression d' Images par Fractales Basée sur la Triangulation de Delaunay* - Grenoble - Institut National Polytecnic de Grenoble – 1995.

[FIS92] FISHER, Yuval - *Fractal Image Compression* – SIGGRAPH, Course Notes - 1992

[FIS95] FISHER, Yuval – *Fractal Image Compression*- Theory and Application, Springer – 1995.

[GHA97] GHARAVI-ALKHANSARI, Mohammad - *Fractal-based image and video coding using matching pursuit* - Illinois - PhD thesis, University of Illinois - 1997.

[GIG97] GÖTTING, D.;IBENTHAL, A.; GRIGAT, R. - *Fractal image coding and magnification using invariant features* - NATO ASI Conf. Fractal Image Encoding and Analysis, Trondheim, July 1995. appears in Fractals, Vol. 5, Supplementary issue - 1997.

[GMS96] GOMES, Jonas; MOTA, Cícero; SILVA, Romildo; VELHO, Luiz -*Image Effects Using Contractive Mappings* - Rio de Janeiro - IMPA-Instituto de Matemática Pura e Aplicada – 1996

[GWO96] GONZALEZ, Rafael ; WOODS, Richard – *Tratamiento digital de imagenes* - Addison Wesley – 1996.

[HAM01] HAMZAOUI, Raouf – *Fractal Image Compression* - Leipzig -, Universität Leipzig / Institut für Informatik – 2001.

[HGS97] HAMZAOUI R.; GANZ, B.; SAUPE, D.- *Quadtree based variable rate oriented mean shape-gain vector quantization* - J. A. Storer, M. Cohn (eds.), Proceedings DCC'97 Data Compression Conference. IEEE Comp. Soc. Press - 1997.

[HMS94] HURTGEN, Bernd; MOLS, Paul; SIMON, Stephan – *Fractal Transform Coding of Color Images* – Institut fur Elektrische-Nachrichtentechnik - SPIE – 1994.

[KLE96] KLEIN, Yaron – *Fractal Image Compression* - M.Sc. Thesis - Tel-Aviv - Tel-Aviv University / Department of Electrical Systems –1996.

[KOM95a] KOMINEK, John - *Advances in Fractal Compression for Multimedia Applications* - Department of Computer Science University of Waterloo, Ontario, Canada – 1995.

[KOM95b] KOMINEK, Jonh - *Algorithm for fast fractal image compression* - Proceedings from IS&T/SPIE 1995 Symposium on Electronic Imaging: Science & Technology, Vol. 2419 Digital Video Compression: Algorithms and Technologies – 1995.

[NOV93] NOVAK, Miroslav - *Attractor Coding of Images* - Department of Electrical Engineering Linköping University – 1993.

[OLI95] OLIVEIRA, Luís Fernando – *Correcção Gama, Sinais, Diferença de Cor e Sistemas de Vídeo* – Rio de Janeiro - LCG-UFRJ – 1995.

[OMD98] OLIVEIRA, J. F. L.; MENDONCA, G.V.; DIAS, R.J. - *A modified fractal transformation to improve the quality of fractal coded images* - Chicago - Proc. ICIP-98 IEEE International Conference on Image Processing - 1998.

[POY99] POYNTON, Charles – *Frequently Asked Questions about Color* – 1999.

[PVR95] PULCINI, Giovambattista; Verrando, Valério ; Rossi, Riccardo; Meloni, Carlo - *IFSAF Theory and Applications* - 1995.

[RHS97] RUHL, M.; HARTENSTEIN, H.; SAUPE, D. - *Adaptive partitionings for fractal image compression* - Proc. ICIP-97 IEEE International Conference on Image Processing, Santa Barbara, California - 1997.

[SCR99] SCURI ,Antonio Escaño - *Fundamentos da Imagem Digital* - Rio de Janeiro - Tecgraf/PUC – 1999.

[SHH96] SAUPE, D.; HAMZAOUI, R.; HARTENSTEIN, H. - *Fractal image compression - an introductory overview* - D. Saupe, J. Hart (eds.), Fractal Models for Image Synthesis, Compression and Analysis, ACM SIGGRAPH'96 Course Notes 27, New Orleans, Louisiana - 1996.

[TEX97] TEXAS INSTRUMENTS, Inc. - *An Introduction to Fractal Image Compression* - Texas Instruments Europe – 1997.

[VRS96] VRSCAY, Edward R.- *A Hitchhiker's Guide to "Fractal-Based" Function Approximation and Image Compression* - Ontario - Department of Applied Mathematics - University of Waterloo - 1996.

[WJA99] WOHLBERG, B. E. ; JAGER, G. - *A review of the fractal image coding literature* - IEEE Transactions on Image Processing, 8(12) - 1999.

[ZPO95] ZHANG, Ying; PO, Lai Man – *Fractal Color Image Compression Using Vectror Distorcion Measure* – Hong Kong - D.E.E. University of Hong Kong- 1995.

Recursos na Internet

[www 1] http://www.caa.uff.br/~aconci/compressao/fractal.htm

[www 2] http://www2.ncsu.edu:8010/scivis/lessons/colormodels/color_models2.html

[www 3] http://inls.ucsd.edu/y/Fractals/

[www 4] ftp://shear.informatik.uni-leipzig.de/pub/Fractal/

[www 5] ftp://links.uwaterloo.ca/pub/Fractals/