Instituto de Computação
Universidade Estadual de Campinas

Codificação de Vídeo
Baseada em Fractais e Representações Esparsas

Vitor de Lima

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA POR
MARIA FABIANA BEZERRA MULLER - CRB8/6162
BIBLIOTECA DO INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E
COMPUTAÇÃO CIENTÍFICA - UNICAMP

Lima, Vitor de, 1985-
L628c Codificação de vídeo baseada em fractais e representações esparsas / Vitor de Lima. – Campinas, SP : [s.n.], 2012.

Orientador: Hélio Pedrini.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Compressão de imagens. 2. Fractais. 3. Método de decomposição. I. Pedrini, Hélio, 1963-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em inglês: Video coding based on fractals and sparse representations
Palavras-chave em inglês:
Image compression
Fractals
Decomposition method
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Hélio Pedrini [Orientador]
João Paulo Papa
Anderson de Rezende Rocha
Data de defesa: 12-03-2012
Programa de Pós-Graduação: Ciência da Computação

TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 12 de março de 2012, pela Banca examinadora composta pelos Professores Doutores:

João Paulo Papa

Prof. Dr. João Paulo Papa
FC / UNESP-Bauru

[illegible]

Prof. Dr. Anderson de Rezende Rocha
IC / UNICAMP

Hélio Pedrini

Prof. Dr. Hélio Pedrini
IC / UNICAMP

Instituto de Computação
Universidade Estadual de Campinas

Codificação de Vídeo
Baseada em Fractais e Representações Esparsas

Vitor de Lima[1]

Março de 2012

[1]Suporte financeiro de: Bolsa do CNPq (processo 132166/2010-0) 2010, Projeto FAPESP (processo 2010/02910-6) 2010–2011 e Universidade Estadual de Campinas 2010–2011.

Resumo

Vídeos são sequências de imagens estáticas representando cenas em movimento. Transmitir e armazenar essas imagens sem nenhum tipo de pré-processamento necessitaria de enormes larguras de banda nos canais de comunicação e uma quantidade massiva de espaço de armazenamento. A fim de reduzir o número de bits necessários para tais dados, foram criados métodos de compressão com perda. Esses métodos geralmente consistem em um codificador e um decodificador, tal que o codificador gera uma sequência de bits que representa uma aproximação razoável do vídeo através de um formato pré-especificado e o decodificador lê essa sequência, convertendo-a novamente em uma série de imagens.

A transmissão de vídeos sob restrições extremas de largura de banda tem aplicações importantes como videoconferências e circuitos fechados de televisão. Neste trabalho são abordados dois métodos destinados a essa aplicação, decomposição usando representações esparsas e compressão fractal.

A ampla maioria dos codificadores tem como mecanismo principal o uso de transformações inversíveis capazes de representar imagens espacialmente suaves com poucos coeficientes não-nulos. Representações esparsas são uma generalização dessa ideia, em que a transformação tem como base um conjunto cujo número de elementos excede a dimensão do espaço vetorial onde ela opera. A projeção dos dados pode ser feita a partir de uma heurística rápida chamada *Matching Pursuit*. Uma abordagem combinando essa heurística com um algoritmo para gerar a base sobrecompleta por aprendizado de máquina é apresentada.

Codificadores fractais representam uma aproximação da imagem como um sistema de funções iterativas. Para isso, criam e transmitem uma sequência de comandos, chamada colagem, capazes de obter uma representação da imagem na escala original dada a mesma imagem em uma escala reduzida. A colagem é criada de tal forma que, se aplicada a uma imagem inicial qualquer repetidas vezes, reduzindo sua escala antes de toda iteração, converge em uma aproximação da imagem codificada.

Métodos simplificados e rápidos para a criação da colagem e uma generalização desses métodos para a compressão de vídeos são apresentados. Ao invés de construir a colagem tentando mapear qualquer bloco da escala reduzida na escala original, apenas um conjunto pequeno de blocos é considerado.

O método de compressão proposto para vídeos agrupa um conjunto de quadros consecutivos do vídeo em um fractal volumétrico. A colagem mapeia blocos tridimensionais entre as escalas, considerando uma escala menor tanto no tempo quanto no espaço. Uma adaptação desse método para canais de comunicação cuja largura de banda é instável também é proposta.

Abstract

A video is a sequence of still images representing scenes in motion. A video is a sequence of extremely similar images separated by abrupt changes in their content. If these images were transmitted and stored without any kind of preprocessing, this would require a massive amount of storage space and communication channels with very high bandwidths. Lossy compression methods were created in order to reduce the number of bits used to represent this kind of data. These methods generally consist in an encoder and a decoder, where the encoder generates a sequence of bits that represents an acceptable approximation of the video using a certain predefined format and the decoder reads this sequence, converting it back into a series of images.

Transmitting videos under extremely limited bandwidth has important applications in video conferences or closed-circuit television systems. Two different approaches are explored in this work, decomposition based on sparse representations and fractal coding.

Most video coders are based on invertible transforms capable of representing spatially smooth images with few non-zero coefficients. Sparse representations are a generalization of this idea using a transform that has an overcomplete dictionary as a basis. Overcomplete dictionaries are sets with more elements in it than the dimension of the vector space in which the transform operates. The data can be projected into this basis using a fast heuristic called Matching Pursuits. A video encoder combining this fast heuristic with a machine learning algorithm capable of constructing the overcomplete dictionary is proposed.

Fractal encoders represent an approximation of the image through an iterated function system. In order to do that, a sequence of instructions, called a collage, is created and transmitted. The collage can construct an approximation of the original image given a smaller scale version of it. It is created in such a way that, when applied to any initial image several times, contracting it before each iteration, it converges into an approximation of the encoded image.

Simplier and faster methods for creating a collage and a generalization of these methods to video compression are presented. Instead of constructing a collage by matching any block from the smaller scale to the original one, a small subset of possible matches is considered.

The proposed video encoding method creates groups of consecutive frames which are used to construct a volumetric fractal. The collage maps tridimensional blocks between the different scales, using a smaller scale in both space and time. An improved version of this algorithm designed for communication channels with variable bandwidth is presented.

Agradecimentos

Eu gostaria de agradecer ao professor Hélio Pedrini pela ajuda tanto na escolha do tema desta dissertação quanto no desenvolvimento dele e ao William Robson Schwartz, co-autor da maioria dos artigos aqui apresentados. Ao CNPq, FAPESP e CAPES pelo apoio financeiro, aos professores, funcionários e amigos do Instituto de Computação e da Universidade Estadual de Campinas e a minha família.

Sumário

Resumo **v**

Abstract **vii**

Agradecimentos **ix**

1 Introdução **1**

1.1 Codificação Fractal de Imagem e Vídeo 2
1.2 Representação Esparsa 2
1.3 Conceitos Adicionais 3
1.4 Organização do Texto 8

2 A Very Low Bit-Rate Minimalist Video Encoder Based on Matching Pursuits **9**

2.1 Prólogo 9
2.2 Abstract 10
2.3 Introduction 10
2.4 Related Work 11
2.4.1 Matching Pursuits 11
2.4.2 Optimized Orthogonal Matching Pursuits 12
2.4.3 K-SVD 12
2.4.4 Matching Pursuit Video Coding 13
2.5 Proposed Video Codec 14
2.6 Experimental Results 14
2.7 Conclusions and Future Work 16
2.8 Acknowledgments 17

2.9 References 17

3 Fractal Image Encoding Using a Constant Size Domain Pool 21

3.1 Prólogo 21
3.2 Abstract 24
3.3 Introduction 24
3.4 Background 25
3.4.1 Fractal Coding 25
3.4.2 Searchless Fractal Coding 26
3.5 Proposed Method 27
3.6 Experimental Results 30
3.7 Conclusions and Future Work 31
3.8 References 32

4 Fast Low Bit-Rate 3D Searchless Fractal Video Encoding 37

4.1 Prólogo 37
4.2 Abstract 38
4.3 Introduction 38
4.4 Background 39
4.4.1 Fractal Image Coding 39
4.4.2 Searchless Fractal Image Coding 40
4.4.3 Fractal Video Coding 42
4.4.4 Fast Rate-Distortion Heuristic 42
4.4.5 Structural Dissimilarity 43
4.5 Proposed Method 45
4.6 Experimental Results 47
4.7 Conclusions and Future Work 50
4.8 References 50

5 Fast Adaptive Transcoding Based on a 3D Low Bit Rate Fractal Video Encoder 57

5.1 Prólogo 57
5.2 Abstract 58
5.3 Introduction 58
5.4 Background 59

5.4.1 Fractal Image Encoding 59
5.4.2 Searchless Fractal Encoding 60
5.4.3 Domain Pool with Constant Size 61
5.4.4 Fractal Video Encoding 61
5.4.5 Searchless Fractal Video Encoding 62
5.4.6 Rate-Distortion Heuristic for Fractal Encoding 63
5.4.7 Structural Dissimilarity 64
5.5 Proposed Method 66
5.5.1 Rate-Distortion Heuristic 67
5.5.2 Fast Fractal Block Video Encoding 68
5.6 Experimental Results 70
5.7 Conclusions and Future Work 73
5.8 References 74

6 Conclusões 80

Referências Bibliográficas 82

Lista de Tabelas

2.1 Average PSNR (in decibels) obtained by using the proposed codec (MP) and H.263. 15

3.1 Quantizers applied according to the area of the range block. 29
3.2 Average encoding and decoding time in miliseconds of the proposed and the searchless method for the benchmark images. 30
3.3 Number of range blocks generated at 0.20 bpp for each tested image. 31

4.1 Quantizers applied according to the volume of the range block. 47
4.2 Number of frames for the used video sequences. 48

5.1 Uniform quantizers applied according to the volume of the range block. 69
5.2 The number of frames for the video sequences, the encoding time of the proposed method and the size of each encoded file with and without the extra data used by the transcoding process. 71

Lista de Figuras

2.1 Comparison of per-frame PSNR values between the proposed encoder and H.263 for Akiyo sequence at 50 kbps. 16
2.2 Comparison of per-frame PSNR values between the proposed encoder and H.263 for Salesman sequence at 50 kbps. 17
2.3 Comparison of per-frame PSNR values between the proposed encoder and H.263 for Hall monitor sequence at 50 kbps. 18

3.1 Comparação entre a abordagem proposta e o método sem busca a 0.08 bpp. 22
3.2 Comparação entre a abordagem proposta e o método sem busca a 0.08 bpp. 23
3.3 Relationship between range (R) and domain (D) blocks in a searchless fractal encoder. 26
3.4 All possible relationships between the range (R) and the domain blocks (D) in the proposed encoder. 28
3.5 The prediction rule used to encode the mean values. 29
3.6 Rate-distortion curve for a number of tested images. 35
3.7 Rate-distortion curve for a number of tested images. 36

4.1 Relationship between range (R) and domain (D) blocks in a searchless fractal encoder. 40
4.2 An example image subdivided by a recursive binary tree partition. . . 41
4.3 Several distorted versions of the same image with different perceptual qualities and approximately the same MSE. 44
4.4 An example of the relationship between range and domain blocks in the proposed 3D encoder. 46

4.5 Differences in the accuracy of frame prediction in both methods at 50 kbps. 49
4.6 Mean structural dissimilarity at different rates for the proposed video encoder and the x264 encoder. 51
4.7 Mean structural dissimilarity at different rates for the proposed video encoder and the x264 encoder. 52
4.8 Mean structural dissimilarity at different rates for the proposed video encoder and the x264 encoder. 53
4.9 Encoding time at different rates for the proposed video encoder and the x264 encoder. 54

5.1 All possible relationships between the range (R) and the domain blocks (D) in the encoder proposed by Lima et al. [31]. 62
5.2 The relationship between the range (R) and the domain block (D) used in the searchless fractal video encoder. 63
5.3 Several distorted versions of the same image with different perceptual qualities and approximately the same MSE. 66
5.4 A few possible relationships between a range and a domain block in the proposed video encoder. 68
5.5 Mean structural dissimilarity at different rates for the proposed video encoder and the x264 encoder. 72
5.6 Mean structural dissimilarity at different rates for the proposed video encoder and the x264 encoder. 73
5.7 Encoding time at different rates for the video transcoder and the x264 encoder. 74
5.8 Encoding time at different rates for the video transcoder and the x264 encoder. 75
5.9 Requested and obtained rates for the proposed method. 76
5.10 Requested and obtained rates for the proposed method. 77

Capítulo 1

Introdução

Um vídeo colorido com resolução de 352×288 pixels a 30 fps (quadros por segundo) requer uma largura de banda de aproximadamente 36.5 Mbits por segundo, se transmitido sem nenhum pré-processamento, um valor elevado para uma resolução tão baixa. Assim, foram criados métodos para transmitir versões aproximadas e mais simples de sequências de imagens, reduzindo a largura de banda necessária para até menos de 1% dos requisitos originais.

Esses métodos realizam a chamada compressão com perda, que consiste em aproximar os dados originais segundo um modelo simplificado. O processo capaz de encontrar parâmetros para esse modelo que aproximam uma determinada sequência de imagens é chamado de codificação, sendo que estes são ajustados a fim de satisfazer uma restrição seja no tamanho total em bits ocupados por eles, seja na qualidade resultante da aproximação fornecida pelo modelo. O uso desses parâmetros no modelo, a fim de obter a aproximação dos dados originais, é chamado de decodificação. Após a codificação, somente os parâmetros desse modelo são transmitidos ao invés da sequência original.

Neste texto foram abordados dois modelos usados na aproximação dos dados originais, os fractais IFS (descritos na Seção 1.1) e as representações esparsas (descritas na Seção 1.2).

1.1 Codificação Fractal de Imagem e Vídeo

Os codificadores fractais constroem um sistema de funções iterativas que descreve uma aproximação da imagem desejada [15]. Para isso, fazem o casamento entre blocos semelhantes de duas escalas distintas da mesma imagem. Cada casamento está sujeito a algumas transformações tanto na geometria dos blocos quanto em seus pixels.

A união desses casamentos é chamada de colagem, que é construída particionando-se a imagem original em blocos (chamados blocos moldes) segundo uma regra pré-definida e procurando para cada bloco particionado na escala reduzida outro bloco (chamado bloco domínio) com o mesmo tamanho e o mais semelhante possível. A semelhança é medida após o bloco domínio passar pelas transformações em sua geometria e em seus níveis de cinza.

O conjunto dos blocos da escala reduzida é chamado de *domain pool*, eles são usados como candidatos a serem casados com os blocos molde. Em casos em que esse conjunto é muito grande por permitir o casamento de um bloco molde em qualquer outra região da imagem, torna-se necessária a criação de heurísticas para evitar buscas exaustivas lentas. A divisão da imagem é feita por meio de estruturas de subdivisão do espaço baseadas em árvores semelhantes às *quadtrees* [46].

A decodificação consiste em criar uma imagem inicial qualquer e, repetidas vezes, aplicar a colagem, reduzir a imagem, aplicá-la novamente até atingir um ponto fixo. Algumas melhorias a esse processo são usadas neste texto a fim de acelerar a convergência.

A generalização usada neste texto para a codificação de vídeo consiste em operar um fractal tridimensional composto por um conjunto de quadros consecutivos [28]. A colagem realiza o casamento entre blocos moldes e blocos domínios volumétricos e tanto o processo de codificação quanto o de decodificação são similares ao caso bidimensional.

1.2 Representação Esparsa

A decomposição sobrecompleta de sinais baseia-se em representar um sinal como uma combinação linear de elementos pertencentes a um conjunto chamado dicionário sobrecompleto. A quantidade de elementos desse conjunto excede a dimensão do espaço vetorial em que a transformação opera, implicando várias representações possíveis para um mesmo sinal. As representações mais esparsas dentro de uma certa margem de erro de representação são as mais úteis nos processos de compressão, entretanto, encontrar a representação mais esparsa é um problema NP-difícil [9].

Logo, muitas aplicações usam heurísticas para encontrar uma solução aceitável para a decomposição, sendo uma delas *Matching Pursuits* [32] e outras heurísticas derivadas desta [41]. Trata-se de uma heurística gulosa que projeta o dado em todos os elementos do dicionário e incorpora o elemento que causa a maior redução no erro de representação. Esse passo é repetido até que o erro total esteja dentro do desejado.

O método proposto particiona os quadros em blocos de 16×16 pixels, codificando cada um deles por meio da heurística *Optimized Orthogonal Matching Pursuit* [41]. O primeiro quadro é codificado por inteiro, enquanto nos quadros subsequentes é codificada a diferença entre eles e seu predecessor. O dicionário sobrecompleto usado nas decomposições foi criado por uma generalização do algoritmo *k-means*, chamada K-SVD [1].

1.3 Conceitos Adicionais

Representar o sinal usando aproximações mais simples é apenas uma parte do processo de codificação. Os coeficientes e estruturas resultantes dessa aproximação precisam ser transmitidos, porém, ainda possuem algumas redundâncias estatísticas. As técnicas apresentadas nesta seção foram criadas a fim de explorar as correlações entre os dados e a entropia deles a fim de reduzir ainda mais a quantidade de bits necessária.

Codificação Aritmética

Dada uma sequência de ocorrências de símbolos, cada uma delas assumindo n possibilidades distintas (chamadas de s_i), a entropia de Shannon [50] demonstra um limite inferior H de bits necessários para transmitir cada ocorrência, definida como

$$H = -\sum_{i=0}^{n-1} p(s_i) \log_2 p(s_i) \qquad (1.1)$$

em que p é uma função de probabilidade da ocorrência de cada símbolo.

Um método conhecido para representar esse tipo de sequência foi proposto por Huffman [22], porém, a principal restrição dessa abordagem é o uso de um número inteiro de bits para codificar cada símbolo, ou seja, em vários casos onde o termo $\log_2 p(s_i)$ não é inteiro, há uma diferença entre o limite inferior H e o tamanho obtido por esta codificação.

A codificação aritmética [43] representa toda a sequência como um único número de tamanho variável. De acordo com a codificação desse número é possível atingir um tamanho extremamente próximo à entropia, entretanto, a um custo computacional maior se comparado à codificação de Huffman.

O algoritmo para a construção desse número começa delimitando um valor máximo e um mínimo para ele. A partir de então, o algoritmo procede recursivamente da seguinte forma: a cada ocorrência a ser codificada, o intervalo entre o máximo e o mínimo é dividido em subintervalos, cada um deles tem seu tamanho (ou seja, a distância entre seus extremos) proporcional a cada $p(s_i)$. O intervalo correspondente à ocorrência atual é escolhido e seus extremos são usados na codificação da próxima ocorrência. Detalhes da implementação de um codificador aritmético rápido podem ser vistos em [45].

Codificação de Golomb-Rice

A codificação de Golomb [19] representa números inteiros não-negativos sem conhecimento prévio de um limite superior para o seu valor. Um número N é codificado utilizando-se um parâmetro M para dividi-lo em duas partes: q e r, em que q é o resultado da divisão de N por M, sendo usualmente representado por codificação unária (isto é,uma sequência de uns seguida de um zero), e r é o resto da divisão de N por M, tal que seu valor está limitado entre 0 e $M-1$ e pode ser codificado por métodos que exigem um limite superior, como a codificação binária ou aritmética.

Valores que não são exclusivamente positivos ou nulos podem ser mapeados pela função T, cuja imagem é o conjunto dos inteiros não-negativos, sendo

$$T(v) = \begin{cases} 2v & \text{se} \quad v \geq 0 \\ -2v-1 & \text{se} \quad v < 0 \end{cases} \tag{1.2}$$

No caso em que M é uma potência de 2, tem-se a codificação de Golomb-Rice [42]. Nela, q é representado pela codificação unária e r pela codificação binária usando $\log_2 M$ bits.

Codificação Adaptativa

Codificadores usualmente possuem parâmetros a serem ajustados (por exemplo, histogramas da ocorrência de símbolos no caso dos codificadores aritméticos e o parâmetro M no caso da codificação Golomb). A eficiência da transmissão dos dados depende fortemente do ajuste desses parâmetros. Assim, muitos métodos envolvem a atualização periódica deles baseada nos dados transmitidos.

A regra de atualização varia conforme a complexidade do método, sendo menos frequente conforme mais elaborada a codificação. Codificadores simples, como o Goulomb-Rice, podem ser atualizados a cada símbolo recebido (por exemplo, mantendo o parâmetro M como a média de todos os dados anteriores como feito no LOCO-I [58]), porém, a manutenção dos histogramas usados pela codificação aritmética é custosa. Logo, regras complexas de atualização foram propostas [45].

Codificação Contextual

Além de atualizar periodicamente os parâmetros de cada codificador, também é possível manter um conjunto de codificadores. Cada codificador é escolhido de acordo com algum critério extraído dos dados transmitidos. Esse critério e seu respectivo codificador tem o nome de contexto.

Assim como no caso da codificação adaptativa, a quantidade de contextos depende da complexidade de cada codificador. No caso do LOCO-I [58], o uso do Golomb-Rice permite a existência de uma quantidade enorme de codificadores, enquanto a quantidade de contextos adaptativos no caso de codificadores aritméticos nunca passa das dezenas.

Predição e Resíduo

A transmissão de dados muitas vezes requer a representação de coeficientes com vários valores possíves (por exemplo, a média de alguma subregião de uma imagem). A codificação de algo que pode assumir tantas possibilidades necessita de cuidado dado que, dependendo da probabilidade de cada valor, a entropia é muito alta.

Para reduzir a entropia desse tipo de dado, preditores são usados. Eles são funções cujo domínio é um subconjunto dos dados transmitidos e a imagem é uma boa estimativa do valor do dado sendo transmitido.

Um exemplo de predição apresentado no LOCO-I [58] é percorrer a imagem da esquerda para a direita e de cima para baixo, usando a vizinhança de cada pixel sendo transmitido para ajustar um plano tridimensional. Esse plano tridimensional é usado como predição do valor deste pixel.

A diferença entre a predição e o valor real dos dados é chamada de resíduo. A transmissão do resíduo apresenta vários problemas, como definir quais seus valores máximo e mínimo e o codificador a ser usado.

Heurística de Controle de Qualidade e Tamanho

Codificar um vídeo ou imagem consiste em representar esses sinais usando algum modelo cujos parâmetros ocupam uma certa quantidade de *bits*. Geralmente, a representação obtida por esse modelo pode ser refinada ou simplificada conforme a quantidade de parâmetros e a precisão destes, formando assim uma relação entre tamanho total usado pelo modelo e a distorção entre o resultado da representação com a imagem original.

É importante controlar o processo de codificação, seja respeitando uma restrição de tamanho e tentando obter a menor distorção possível ou respeitando uma restrição de distorção minimizando o tamanho. Vários métodos com diferentes complexidades existem para conseguir isso, incluindo alguns baseados em otimização [39].

Os métodos baseados na codificação fractal precisam subdividir a imagem em blocos. Assim, quanto mais densa a subdivisão, maior o tamanho dos parâmetros do modelo. Alguns codificadores usam a heurística de limiarização [16, 59]. Ela consiste em dividir a imagem em uma grade uniforme de blocos e subdividir recursivamente cada um deles até que a distorção destes esteja abaixo de um limiar pré-determinado. Ou seja, esse algoritmo é capaz de controlar a distorção, entretanto, não limita nem minimiza o tamanho.

Outra heurística [47] divide a imagem em uma grade uniforme, inserindo todos os blocos resultantes em uma fila de prioridade ordenados pela distorção. A representação da imagem é incrementalmente refinada a cada passo do algoritmo por meio da subdivisão do bloco com a maior distorção. É possível controlar o tamanho da solução, pois este aumenta apenas um pouco a cada passo. Não há garantia de que esse processo obtém uma solução ótima, pois para tanto seria necessário o uso de otimização, algo com custo computacional bastante elevado.

Transformada de Níveis de Cinza

Obter uma aproximação de um bloco molde a partir de um bloco domínio requer o uso de uma transformada que altere os níveis de cinza. Uma delas, proposta por Øien e Lepsøy [38], é dada por

$$G(D) = \alpha(D - \bar{D}J) + \bar{r}I \tag{1.3}$$

em que G é a transformada de níveis de cinza, D é o bloco domínio subamostrado, $\bar{D}$ e $\bar{r}$ são as médias dos blocos domínio e molde, respectivamente. J denota uma matriz preenchida somente com valores iguais a 1 e com as mesmas dimensões do bloco molde e α é o parâmetro de escala.

A transformada proposta anteriormente por Jacquin [25] tentava determinar uma constante aditiva e um coeficiente multiplicativo para o bloco domínio através de mínimos quadrados. Ao comparar essas duas transformadas, a Equação 1.3 possui várias vantagens que aceleram a convergência. A média do bloco domínio é calculada durante o processo de decodificação ao invés de ser parte dos coeficientes da transformada e a média do bloco molde é forçada ao seu valor correto já na primeira iteração, como detalhado por Pi et al. [40]. É possível construir uma imagem preenchida apenas com as médias dos blocos moldes e usá-la como imagem inicial na decodificação [34], reduzindo ainda mais o número de iterações necessárias e evitando o aparecimento de artefatos.

Decodificação Rápida de Fractais

Um decodificador fractal usa duas imagens, uma versão contraída do sinal onde estão os blocos domínios e outra versão na escala original onde ficam os blocos molde. A cada iteração, a colagem é aplicada e a imagem resultante é contraída para ser usada no próximo passo.

O método proposto por Hamzaoui [21] usa apenas uma imagem a fim de economizar espaço de memória e acelerar a convergência. O conteúdo de cada bloco molde é sobrescrito com seu respectivo domínio contraído e transformado, assim os blocos moldes vizinhos são afetados por essa alteração durante a mesma iteração, não sendo necessário esperar até que ocorra a contração da imagem toda, como no outro método.

Transcodificação

A transcodificação [24] é um processo pelo qual o formato de transmissão, a resolução (seja temporal ou espacial), o tamanho em bits ou a qualidade de um vídeo são alterados sem a necessidade de se recodificar o sinal, trabalhando somente com o sinal comprimido. Neste texto é descrito um método para a alteração do tamanho total em bits de um vídeo a fim de se obedecer uma certa restrição na largura de banda do canal de comunicação usado na transmissão.

1.4 Organização do Texto

Esta dissertação está organizada como segue. Os próximos quatro capítulos apresentam os artigos que foram publicados ou submetidos durante o período de vigência do mestrado. Finalmente, o último capítulo apresenta as conclusões do trabalho.

Capítulo 2

A Very Low Bit-Rate Minimalist Video Encoder Based on Matching Pursuits

2.1 Prólogo

O artigo [8], apresentado nesta seção, foi parte do *15th Iberoamerican Congress on Pattern Recognition (CIARP'2010)* e publicado pela *Springer-Verlag*.

Nele é apresentado um método baseado na representação sobrecompleta de sub-blocos com 16×16 pixels de cada quadro do vídeo, uma abordagem simplificada semelhante à encontrada no padrão JPEG. Cada bloco é decomposto como uma combinação linear de elementos do dicionário sobrecompleto criado por Aharon et al. [1], obtido pela aplicação do algoritmo K-SVD em um conjunto de imagens naturais.

Esse dicionário é não-separável, ou seja, os elementos não são o produto externo de dois vetores unidimensionais. Isto causa um aumento na complexidade da execução da heurística de *Matching Pursuits*, logo, utilizou-se uma unidade de processamento gráfico a fim de atingir uma velocidade de compressão em tempo real.

A maioria das abordagens anteriores era baseada na convolução de cada elemento do dicionário sobrecompleto com todo o quadro. Além do custo computacional superior, que força o uso de dicionários separáveis, essa convolução adiciona um grau de liberdade a mais a cada termo da decomposição, que é a posição onde o elemento do dicionário precisa ser encaixado, introduzindo um parâmetro difícil de ser comprimido.

A qualidade de imagem nos exemplos testados foi aceitável, sendo comparável ao padrão H.263 nas sequências apresentadas. A complexidade computacional do método, a dificuldade em melhorar a quantização dos coeficientes e a codificação sem perda dos parâmetros gerados dificultaram o progresso dessa pesquisa.

2.2 Abstract

This work proposes and implements a simple and efficient video encoder based on the compression of consecutive frame differences using sparse decomposition through matching pursuits. Despite its minimalist design, the proposed video codec has performance compatible to H.263 video standard and, unlike other encoders based on similar techniques, is capable of encoding videos in real time. Average PSNR and image quality consistency are compared to H.263 using a set of video sequences.

2.3 Introduction

Video compression at very low bit-rates is needed for applications that operate using low bandwidth communication channels, for instance, video transmission in mobile equipments. Some techniques that have been suggested for such applications include hybrid-DCT coding [6], wavelet-based coding [21], model-based coding [2], and fractal coding [12].

Extreme compression rates demanded by low bit-rate video applications require unusual video encoding techniques. One possible approach is the matching-pursuit video coding, however, it involves a very time-consuming encoding process [16] due to its exhaustive image scan in order to find patterns that can be represented efficiently.

The approach proposed in this paper is extremely simple and capable of compressing video sequences in real time. The video encoder compresses only the difference between two consecutive frames through matching pursuits. No motion compensation algorithm [10] is used in the process and the quantization is performed by rounding the coefficients to the nearest integer. An innovation of the proposed method is the subdivision of the frame into blocks and application of matching pursuit to each block instead of scanning the entire image looking for regions that have relevant characteristics that can be compressed and then applying matching pursuits to those regions.

A dictionary generated by K-SVD algorithm [1] is used to create sparse decompositions of the processed frame sub-blocks, which are compressed by a context-adaptive arithmetic encoder [19].

Compared to H.263 video codec [11], which has a motion compensation algorithm, more sophisticated quantizers and mechanisms for rate-distortion optimization, the proposed method achieves compatible PSNR values, as demonstrated in the experiments using well-known benchmark video sequences at several average bit rates per second.

The text is organized as follows. Section 2.4 describes the main algorithms used in the proposed solution, as well as reviews of some relevant encoders based on matching pursuits found in literature. Details of the proposed methodology are presented and discussed in Section 2.5. Experimental results obtained with our video codec are shown in Section 2.6. Finally, conclusions of the work and future directions are presented in Section 2.7.

2.4 Related Work

This section briefly describes some relevant concepts and techniques related to the proposed video encoder.

2.4.1 Matching Pursuits

Transforms, such as DCT [5], decompose signals as a linear combination of mutually orthogonal elements belonging to a predetermined basis. This basis contains a minimum number of elements sufficient to express any vector belonging to a particular vector space.

A possible generalization for such type of transform involves using more than the minimum required number of elements within the basis, thus forming an overcomplete dictionary, In this case, a single vector has several possible decompositions and, for data compression purpose, the most interesting decompositions are those that have the largest possible number of linear coefficients equal to zero.

Finding such decompositions is a NP-hard problem [7], so that matching pursuits [13] is a greedy heuristic for finding a very sparse decomposition of a signal using low processing time. Given an overcomplete dictionary $D = \{g_\gamma\}_{\gamma \in \Gamma}$, a signal f to be decomposed and a threshold of the decomposition error ϵ, Algorithm 1 determines which elements of D and linear coefficients are used in a sparse decomposition of f. Term R^k is the signal residue not yet represented by the chosen bases until step k.

Algorithm 1 Matching pursuit algorithm.

```
$R^0 f = f$
$n = 0$
**repeat**
  $i = \arg\max_{k \in \Gamma} \langle R^n f, g_k \rangle$
  $R^{n+1} = R^n f - \langle R^n f, g_i \rangle g_i$
  $n = n + 1$
**until** $n < n_{max}$ OR $|R^{n+1} f| < \epsilon$
```

2.4.2 Optimized Orthogonal Matching Pursuits

A more powerful heuristic for searching for sparse signal representations using overcomplete dictionaries was employed in the proposed video codec, known as optimized orthogonal matching pursuit [18].

At each step of the encoding process, after choosing an element g_i of the dictionary by the same criterion of the conventional matching pursuit, such search heuristic orthogonalizes the entire dictionary with respect to g_i. Therefore, the chosen element in the following step is orthogonal to all elements used previously. The heuristic ensures more sparse representations at a higher computational cost.

2.4.3 K-SVD

A well generated overcomplete dictionary ensures more sparse decompositions, provides a higher convergence speed in matching pursuits, is capable of representing only psychovisually significant features and ignores minor irrelevant details. It is possible to develop such dictionaries through machine learning algorithms [20], among them the K-SVD, which is a generalization of the algorithm for solving the K-means problem.

Two alternating steps are performed during its execution. In the first step, data from the training set is decomposed according to the initial overcomplete dictionary to be optimized using any algorithm capable of doing it. In the second step, each element of the dictionary is replaced by a new one, calculated to minimize the error of each data from the training set that used it in its sparse decomposition, as described in Algorithm 2.

Algorithm 2 K-SVD algorithm.

```
Input: initial set $Y = \{y_i\}_{i=1}^{N}$ of training signals, an initial dictionary $D$ with
normalized columns, a target sparsity $T$ and the total number of iterations $k$.
Output: an approximate solution to $min_{D,X}\{||Y - DX||_F^2\}$ subject to $\forall i, ||x_i||_0 \leq T$
and $\forall j, ||D_j||_2 = 1$.
for $n = 1$ to $k$ do
  $\forall i, x_i = \arg\min_\gamma \{||y_i - D\gamma||_2^2\}$ subject to $||\gamma||_0 \leq T$
  for each column $j$ in $D$ do
    $D_j = 0$
    $I = \{$indices of the signals in $Y$ whose decompositions use $D_j\}$
    $E = Y_I - DX_I$
    $\{d, g\} = \arg\min_{d,g} ||E - dg^T||_F^2$ subject to $||d||_2 = 1$
    $D_j = d$
    $X_{j,I} = g^T$
  end for
end for
```

2.4.4 Matching Pursuit Video Coding

The absolute majority of video codecs based on matching pursuits [3, 15, 22, 23] have their origins in [16]. The method uses an inner-product search to decompose motion residual signals over an overcomplete dictionary of 2D separable Gabor functions.

Despite the high computational cost of such search, the approach avoids artificial block edges and presents both better perceptual image quality and higher PSNR than DCT-based methods for low bit rates video coding. However, the dictionary must be efficiently built to allow fast inner-product computation between its elements and various regions of the residue.

The proposed encoder avoids performing costly searches working similarly to DCT-based coders, where the difference between two consecutive frames is partitioned into non-overlapping blocks that are independently coded using matching pursuits. This allows encoding parallelization of the sub-blocks, however, it does not prevent artifact appearance at the intra-block edges.

2.5 Proposed Video Codec

Initially, the encoder calculates the difference between the frame to be processed and the previous uncompressed frame. If the norm of this subtraction is greater than a certain threshold, the entire frame is used in the next step, otherwise only the difference between these two frames is used.

The image generated in the previous step is then subdivided into blocks of 8×8 pixels without overlapping. Each block is decomposed as a sparse linear combination of the dictionary elements through the Optimized Orthogonal Matching Pursuit algorithm [18]. The average bit rate is controlled by manually varying the error threshold ϵ used in the algorithm.

The overcomplete dictionary used in our encoding method is the same used by Elad and Aharon [9] for image denoising. The learned dictionary contains 256 elements and was trained using K-SVD algorithm using a number of several photographs as a training set.

In the final step, a flag is coded to indicate whether what is being transmitted is only the difference between two consecutive frames or an entire frame.

For each block of the current frame decomposed in the previous step, its sparse representations are transmitted through an arithmetic encoder using four distinct symbols, each one containing its proper adaptive context. The first symbol indicates the number of elements of the dictionary used in the decomposition of that block. For each used element, sign and magnitude of the linear coefficient associated with that element and its index are transmitted in different symbols.

2.6 Experimental Results

The proposed video codec was implemented on a graphics processing unit (GPU) with CUDA [17]. Our codec was compared to the implementation of the H.263 video standard present in the open-source `libavcodec` library [4].

Several video sequences were used in the experiments [14]. Results for three video samples are reported in this work. The videos have resolution of 176×144 pixels and 10 frames per second with subsampled chrominance (format 4:2:2).

All videos were compressed both with our codec and H.263 at different average rates of kilobits per second. The comparison was based on peak signal-to-noise ratio (PSNR) value, expressed by

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{255^2}{\mathrm{MSE}}\right) \tag{2.1}$$

where MSE is the mean squared error between the resulting image after compression and uncompression steps and the original image.

Average PSNR values for all frames and three color channels of the tested video sequences are shown in Table 2.1.

	Akiyo		Salesman		Hall Monitor	
kbps	H.263	MP	H.263	MP	H.263	MP
15	30.70	30.71	29.41	28.74	29.54	29.07
20	31.67	31.73	30.01	29.38	30.22	30.18
30	33.60	33.38	31.21	30.55	31.60	32.11
40	35.20	34.66	32.29	31.55	32.88	33.56
50	36.54	35.77	33.18	32.30	34.06	34.80

Table 2.1: Average PSNR (in decibels) obtained by using the proposed codec (MP) and H.263.

Despite the extreme simplicity of the proposed approach, its performance is very similar to H.263 video standard. The lack of a motion compensation algorithm prevented effective use of statistical redundancy present in the consecutive video frames.

Another important characteristic of the presented approach is its consistency in the video frame quality. As can be seen in Figures 2.1, 2.2 and 2.3, PSNR value of each frame changed abruptly when compressed by H.263, however, it is kept almost constant by the proposed algorithm. This is mainly due to the rate control mechanism of H.263.

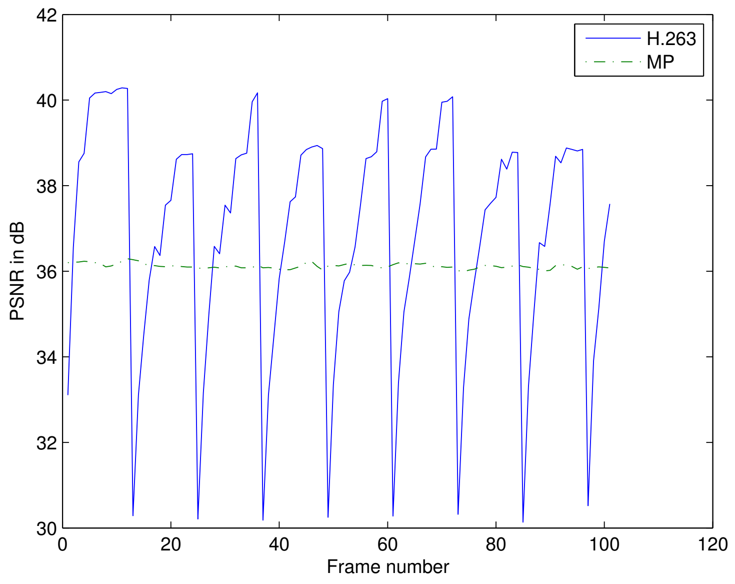


Figure 2.1: Comparison of per-frame PSNR values between the proposed encoder and H.263 for Akiyo sequence at 50 kbps.

2.7 Conclusions and Future Work

A video encoder is proposed to compress the difference between two consecutive frames through the matching pursuit approach using a dictionary previously trained by K-SVD method.

Unlike other video codecs based on matching pursuits, the proposed approach is able to encode video in real time and has performance compatible to H.263 when tested for some video sequences used in standard benchmarks.

Future directions for work include the implementation of refined motion compensation methods, a filter for removing blocking artifacts, a better quantization scheme of the sparse decomposition coefficients and other forms of prediction residue coding using both matching pursuits and dictionaries created by K-SVD. Such changes can significantly improve the resulting image quality.

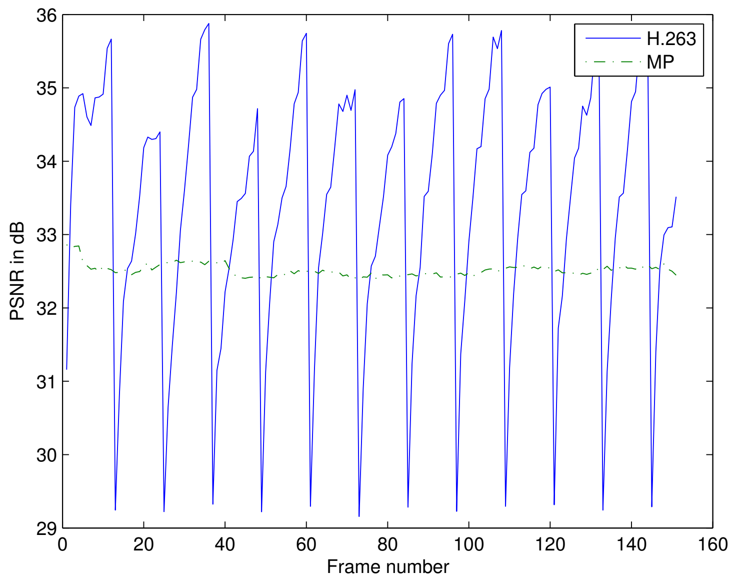


Figure 2.2: Comparison of per-frame PSNR values between the proposed encoder and H.263 for Salesman sequence at 50 kbps.

2.8 Acknowledgments

The authors are grateful to FAPESP and CNPq for their financial support.

2.9 References

[1] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, November 2006.

[2] K. Aizawa, H. Harashima, and T. Saito. Model-Based Analysis Synthesis Image Coding (MBASIC) System for a Person's Face. *Signal Processing: Image Communications*, 1(2):139–152, October 1989.

[3] O.K Al-Shaykh, E. Miloslavsky, T. Nomura, R. Neff, and A. Zakhor. Video Compression using Matching Pursuits. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(1):123–143, February 1999.

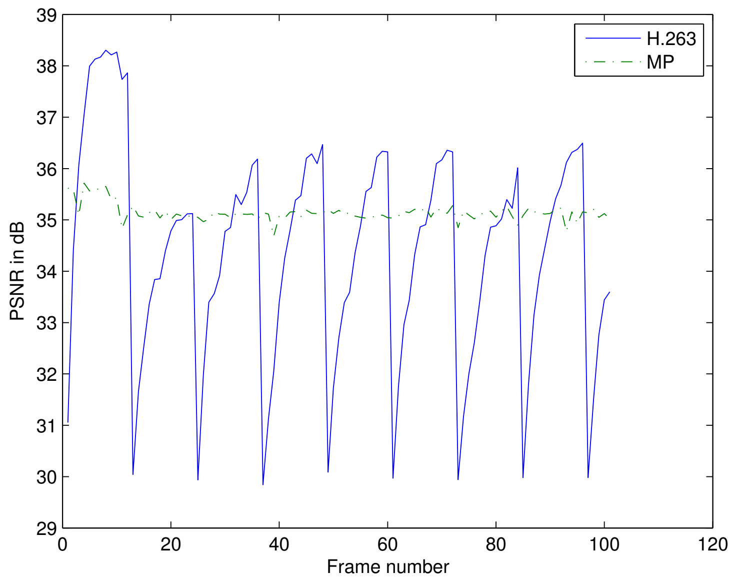


Figure 2.3: Comparison of per-frame PSNR values between the proposed encoder and H.263 for Hall monitor sequence at 50 kbps.

[4] avcodec. libavcodec: A Library containing Decoders and Encoders for Audio/Video Codecs, 2010. `http://www.ffmpeg.org/`.

[5] V. Bhaskaran and K. Konstantinides. *Image and Video Compression Standards: Algorithms and Architectures*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[6] CCITT. Video Codec for Audiovisual Services at p $\times$ 64 kbit/s, CCITT Recommendation H.261, CDM XV-R 37-E, August 1990.

[7] G. Davis. *Adaptive Nonlinear Approximations*. PhD thesis, Department of Mathematics, New York University, 1994.

[8] Vitor De Lima and Helio Pedrini. A Very Low Bit-rate Minimalist Video Encoder Based on Matching Pursuits. In *Proceedings of the 15th Iberoamerican congress conference on Progress in pattern recognition, image analysis, computer vision, and applications*, CIARP'10, pages 176–183, Berlin, Heidelberg, 2010. Springer-Verlag.

[9] M. Elad and M. Aharon. Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745, December 2006.

[10] B. Furht and B. Furht. *Motion Estimation Algorithms for Video Compression*. Kluwer Academic Publishers, Norwell, MA, USA, 1996.

[11] H263. ITU-T Recommendation H.263, Video Coding for Low Bit Rate Communication, September 1997.

[12] A.E. Jacquin. Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations. *IEEE Transactions on Image Processing*, 1(1):18–30, January 1992.

[13] S. Mallat and Z. Zhang. Matching Pursuit with Time-Frequency Dictionaries. *IEEE Transactions on Signal Processing*, 41:3397–3415, December 1993.

[14] Xiph.org Test Media. Video Sequences, 2010. `http://media.xiph.org/video/derf/`.

[15] R. Neff, T. Nomura, and A. Zakhor. Decoder Complexity and Performance Comparison of Matching Pursuit and DCT-based MPEG-4 Video Codecs. In *International Conference on Image Processing*, pages 783–787, Chicago, IL, USA, October 1998.

[16] R. Neff and A. Zakhor. Very-Low Bit-Rate Video Coding Based on Matching Pursuits. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(1):158–171, February 1997.

[17] NVIDIA. CUDA - Parallel Computing Architecture, 2010. `http://www.nvidia.com/`.

[18] L. Rebollo-Neira and D. Lowe. Optimized Orthogonal Matching Pursuit Approach. *IEEE Signal Processing Letters*, 9(4):137–140, April 2002.

[19] A. Said. *Lossless Compression Handbook*, chapter Arithmetic Coding. Communications, Networking, and Multimedia. Academic Press, 2003.

[20] D. Sculley and C.E. Brodley. Compression and Machine Learning: A New Perspective on Feature Space Vectors. In *Data Compression Conference*, pages 332–332, Snowbird, UT, USA, March 2006.

[21] J. Shapiro. Application of the Embedded Wavelet Hierarchical Image Coder to Very Low Bit Rate Image Coding. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 558–561, Minneapolis, MN, USA, April 1993.

[22] B. Wang, Y. Wang, and P. Yin. A Two Pass H.264-Based Matching Pursuit Video Coder. In *IEEE International Conference on Image Processing*, pages 3149–3152, Atlanta, GA, USA, October 2006.

[23] H. Zhang, X. Wang, W. Huo, and D.M. Monro. A Hybrid Video Coder Based on H.264 with Matching Pursuits. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 889–892, Toulouse, France, July 2006.

Capítulo 3

Fractal Image Encoding Using a Constant Size Domain Pool

3.1 Prólogo

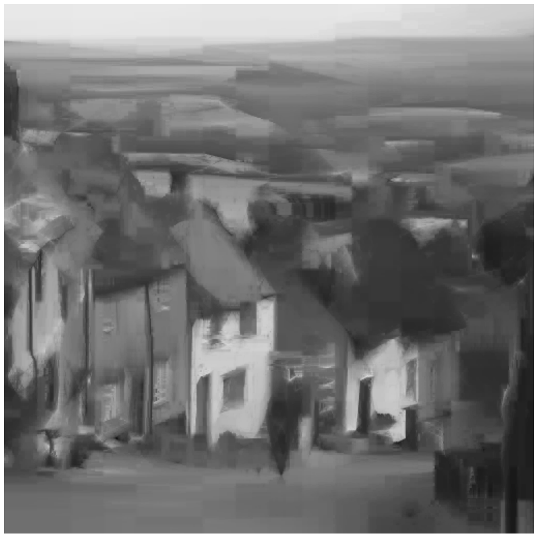
O artigo [6], apresentado nesta seção, foi parte do *VI Workshop de Visão Computacional (WVC'2010)* e publicado nos anais do evento.

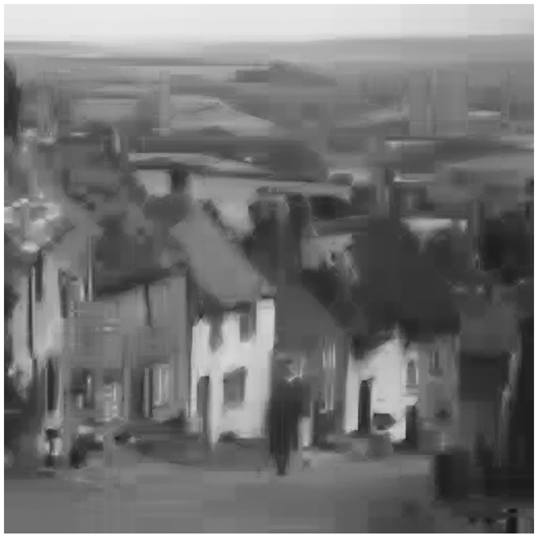
Um novo método de construção de colagens é proposto e comparado com o método sem busca de Furao and Hasegawa [3]. Ao invés de usar apenas uma possibilidade para o casamento entre blocos molde e domínio, uma quantidade constante de 9 candidatos é considerada. Isso implica uma leve melhoria na qualidade de imagem e um aumento considerável no custo computacional, porém, na mesma ordem de grandeza dos métodos rápidos anteriormente propostos.

A estrutura de subdivisão binária foi usada na criação dos blocos moldes em que, ao invés de dividir recursivamente cada região em 4 partes, elas são divididas pela metade na direção vertical ou horizontal, como proposto por Wu et al. [17]. Para o controle de tamanho e distorção, utilizou-se a heurística apresentada na Seção 1.3.

Outras características interessantes da abordagem proposta são o uso da codificação aritmética, o preditor usado nos valores médios dos blocos moldes e os métodos de aceleração da decodificação das Seções 1.3 e 1.3.

As Figuras 3.1 e 3.2 mostram comparações entre o método proposto e o método sem busca. É possível perceber que há artefatos menos impactantes e menos distorção na geometria da cena.


(a) Codificador proposto


(b) Codificador sem busca

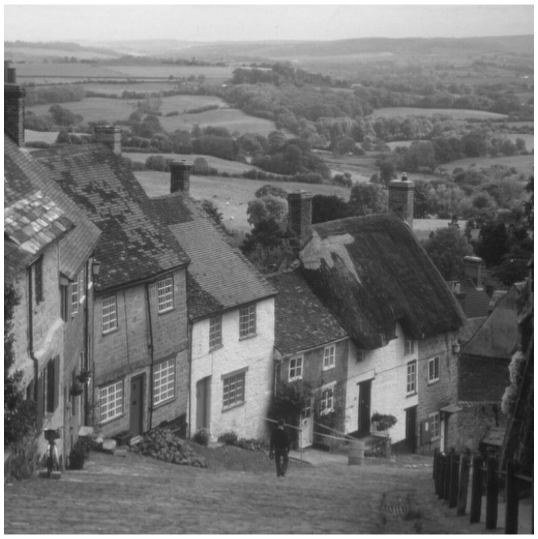
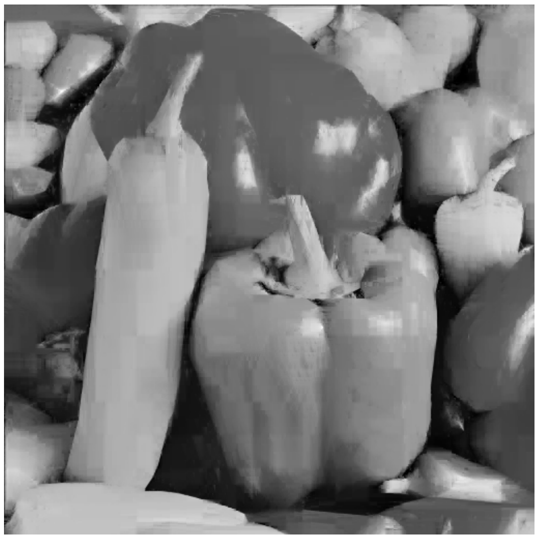
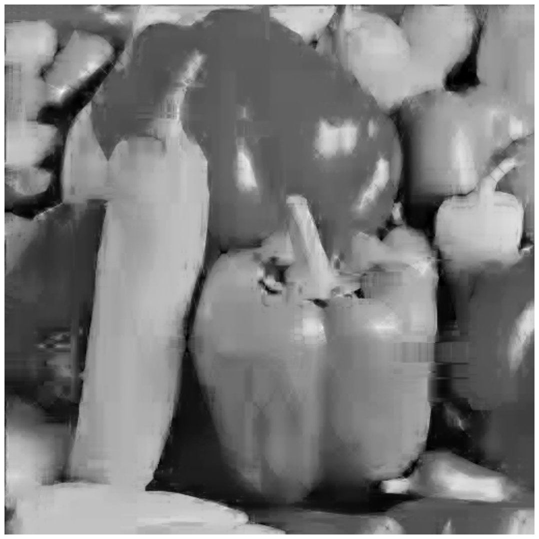

(c) Original

Figura 3.1: Comparação entre a abordagem proposta e o método sem busca a 0.08 bpp.


(a) Codificador proposto


(b) Codificador sem busca

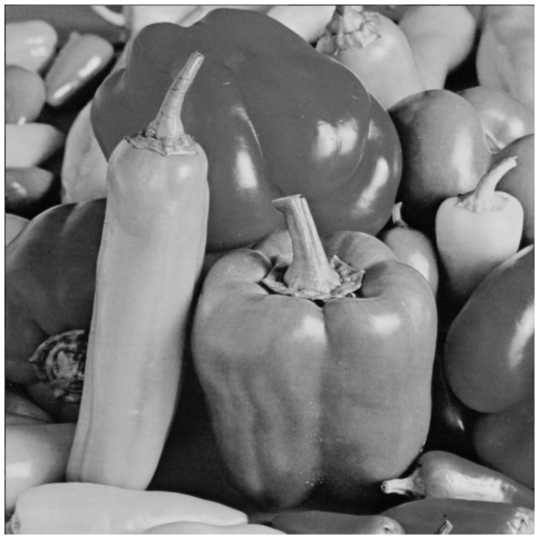

(c) Original

Figura 3.2: Comparação entre a abordagem proposta e o método sem busca a 0.08 bpp.

3.2 Abstract

Image compression techniques play an important role in data reduction and transmission. This work proposes a fractal image encoder based on a small domain pool with constant size constructed from the neighborhood of each range block and an efficient spatial subdivision data structure. This method is compared to a searchless algorithm using well-known grayscale images. The proposed approach is capable of performing fast compression and decompression, while maintaining high visual fidelity and operating at low bit-rates.

3.3 Introduction

Most fractal image encoders are characterized by extremely slow encoding times (taking a few hours to encode a single image) and fast decompression [5]. This is caused by the fact that constructing a fractal to approximate the content of an image is much more complex than displaying it [11]. Unfortunately, they are not competitive with other encoders based on transform coding both in encoding speed and rate-distortion performance.

A fractal encoder outputs a transform called *collage* that, given a downsampled version of the original image, it can create an approximation of the signal in its original scale. The usual method to create this is based on numerous searches for similar blocks between the image in these two scales which can be accelerated by specially designed heuristics [5].

A fast method for constructing a collage, called *searchless fractal compression*, was proposed by Furao and Hasegawa [3]. Each block in the original scale (called *range block*) is associated to a single block in the higher scale (called *domain block*). If the similarity between these two blocks is acceptable, the range block is encoded, otherwise, it is partitioned and the process is applied to the resulting sub-blocks. The encoders based on this concept are faster by orders of magnitude when compared to other fractal encoders employing more complex fractal-based algorithms, taking less than one second to encode an image instead of hours.

This work proposes a fast method for encoding images that is not as restrictive as the searchless encoders while keeping the encoding speed approximately at the same order of magnitude and improving the image quality at similar bit rates.

The next sections are organized as follows. Section 3.4 presents a brief review of fractal coding and searchless methods. The proposed method is described in Section 3.5. Experimental results comparing the proposed method against a searchless encoder are presented in Section 3.6. Finally, the conclusions of the work are given in Section 3.7.

3.4 Background

This section presents a brief review of fractal image coding and the searchless fractal encoders.

3.4.1 Fractal Coding

Fractal image encoders [5] generate a transform as output, called *collage*, that approximates the original image given a downsampled version of it. This transform is created in such way that when applied several times to any image, it will converge to an approximation of the original signal. Based on this property, starting with an arbitrary image, the decoder only needs to apply the collage and downsample its results repeatedly until it converges to the desired output.

Most fractal encoders construct the collage through a subdivision of the image into blocks (range blocks) that are matched against higher scale blocks (domain blocks) in the downsampled image by means of costly searching algorithms. This process makes the encoding process extremely expensive, specially when compared to the decoding time.

The collage can apply any affine transform to map a domain block into a range one. The collage also changes the intensities of the pixels contained in the domain blocks by using a transform such as the one proposed by Øien and Lepsøy [8], shown in Equation 3.1, where G is the resulting pixel intensity, D is the intensity in the domain block, $\bar{D}$ is the mean intensity of the pixels in the domain block, $\bar{r}$ is the mean intensity of the pixels in the range block and α is a parameter determined by a least squares estimation.

$$G(D) = \alpha(D - \bar{D}) + \bar{r} \tag{3.1}$$

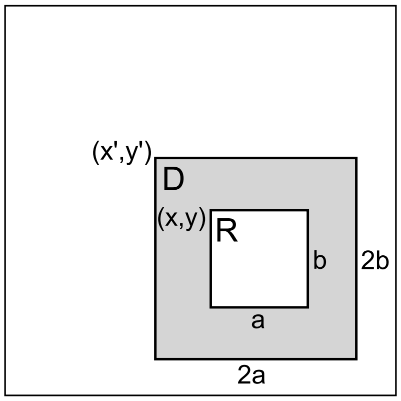


Figure 3.3: Relationship between range (R) and domain (D) blocks in a searchless fractal encoder.

3.4.2 Searchless Fractal Coding

Initially proposed by Furao and Hasegawa [3], the searchless fractal coding aims at quickly constructing a collage avoiding the complex search for the best match between range and domain blocks. Each range block at the coordinates (x, y) with dimensions $a \times b$ is matched against only one domain block with dimensions $2a \times 2b$ located at $(x - a/2, y - b/2)$, as shown in Figure 3.3. To encode a range block, only α and $\bar{r}$ must be transmitted, instead of the large number of parameters used in other state-of-the-art fractal encoders.

The image is partitioned into equally sized blocks using a quadtree, which are encoded by the process previously described. For each block. if the obtained error is larger than a certain threshold, the block is recursively divided into four sub-blocks until a certain minimum size is reached.

The searchless encoder proposed by Furao and Hasegawa [3] was later refined in [17] by employing a more flexible data structure to subdivide the image, in which a range block can be split into half either in the horizontal or vertical directions. There is no limit to the size of the range blocks, but smaller regions always have their α parameter set to zero and $\bar{r}$ is encoded using fewer bits.

3.5 Proposed Method

This paper proposes a new method for encoding the relationship between the domain and range blocks in images that is not as restrictive as the searchless methods, but is not as complex as the classical brute force or heuristic algorithms [2].

A more flexible domain pool with a larger selection of possible domain blocks is created to avoid the necessity of the searchless encoders to split certain regions of the image several times until they reach the desired reconstruction quality. This selection avoids such excessive use of subdivisions and also can reduce the number of bits used to encode the image.

For each range block with dimensions $a \times b$ located at (x, y), there are 9 possible domain blocks that can be used to represent it. The position of these blocks can be expressed as

$$\begin{aligned} x' &= x - a + p_x \times a/2 \\ y' &= y - b + p_y \times b/2 \end{aligned} \tag{3.2}$$

where p_x and p_y must be equal to 0, 1 or 2. All possible domain blocks given by this equation are illustrated in Figure 3.4. These candidate domain blocks have dimensions equal to $2a \times 2b$ and only one of them is chosen as the definitive mapping by testing which one of them has the lowest error when compared to the original range block, after having its gray values properly transformed by Equation 3.1.

If the range block must be split to achieve a lower reconstruction error, then it is divided into two equally sized sub-blocks. However, to do so, it must decide in which direction this subdivision will be performed. The heuristic used to choose how to properly partition a block tries to encode both sub-blocks separately without actually subdividing them, but by evaluating the two possible directions that can be used to split the original block. The direction resulting in the lowest estimated error is chosen.

The encoder initially subdivides the image into a uniform grid of blocks with 64×64 pixels. The rate-distortion heuristic proposed by Saupe et al. [13] is used for rate control, the blocks are inserted into a priority queue to sort the range blocks according to their sum of squared differences (SSD) instead of the mean squared difference (MSE), this replacement was proposed by Fisher and Menlove [1] to improve both the PSNR and the perceptual quality of the resulting image. At each iteration of the decoder, the range block with the largest SSD is subdivided into two blocks, which

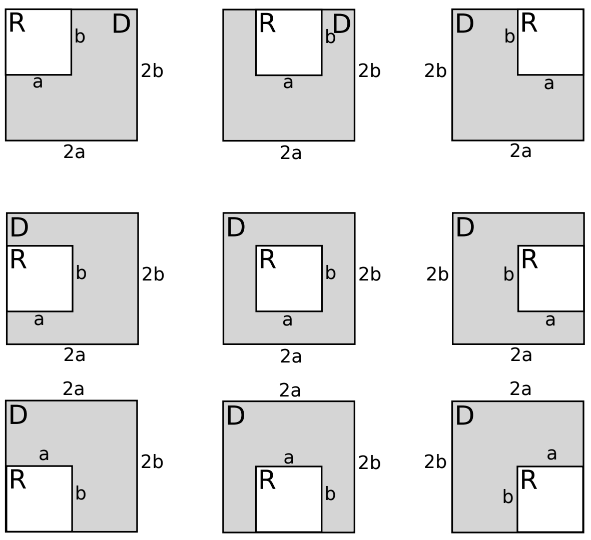


Figure 3.4: All possible relationships between the range (R) and the domain blocks (D) in the proposed encoder.

are reinserted into the queue. The total number of iterations, N_{it}, is defined by the user.

After the last iteration, the range blocks in the queue are transmitted by a context-adaptive arithmetic encoder [12], where α parameters are quantized into one of the values in the set $\{0.25, 0.5, 0.75, 1.0\}$, as suggested by Tong and Pi [16], and encoded by an adaptive context chosen by the $\lfloor \log_2 A \rfloor$, where A is total area of the block, p_x and p_y values are encoded by their own adaptive contexts. Range blocks with only one or two pixels have their α parameter set to zero as suggested by Wu et al. [17].

The parameter $\bar{r}$ is encoded by a prediction method similar to the one proposed by Teuhola [15], in which the range blocks are scanned recursively from the left to the right and from top to bottom. Each visited range block predicts its mean value by averaging four mean values from neighboring blocks (as shown in Figure 3.5), quantizes the prediction and the real mean value using a uniform quantizer and an adaptive context chosen by the $\lfloor \log_2 A \rfloor$, as shown in Table 3.1 to encode the difference between them.

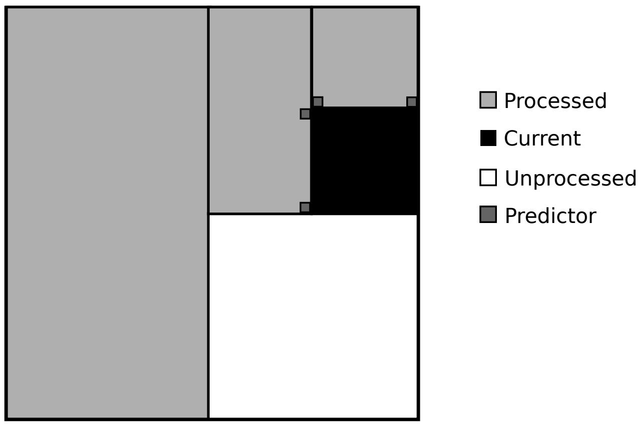


Figure 3.5: The prediction rule used to encode the mean values.

Decision Table for the Quantization of $\bar{r}$		
area	quantization step	number of used bits
1	16	4
2	16	4
4	8	5
8	8	5
16	4	6
32	4	6
64	2	7
≥ 128	1	8

Table 3.1: Quantizers applied according to the area of the range block.

The spatial subdivision binary tree is transmitted by two different symbols: a binary flag for each node to mark it as either a leaf or an internal node, which is encoded by an adaptive context based on the $\lfloor \log_2 A \rfloor$ and another symbol to indicate the direction that node was split, which has two different contexts, one for blocks with b larger than a and another one for the remaining blocks.

The decoding process is accelerated by 3 different and complementary methods. The initial image that is used in the first iteration of the decoder is composed by filling each range block with its mean (for more details, see [7]). The used pixel intensity transform is the one proposed by Øien and Lepsøy [8] with additional proofs and details given by Pi et al. [10]. Each iteration is applied according to the Gauss-Seidel inspired method proposed by Hamzaoui [4], which uses only one image during the iterations overwriting each range block with its updated contents. The use of these methods assures that the decoding process converges in 4 iterations or less, instead of the usual 8 to 10 iterations used by other fractal decoders.

Finally, the image is post-processed by the same deblocking filter proposed by Fisher and Menlove [1] that adapts itself according to the size of the range block, ignoring the ones with smaller areas and using more aggressive parameters in the larger ones.

3.6 Experimental Results

Images	Searchless		Constant-Sized Domain Pool	
	encoding time	decoding time	encoding time	decoding time
Baboon	21.9	19.2	100.6	19.3
Barbara	21.0	19.0	92.3	19.2
Boat	20.3	18.8	94.5	19.2
Goldhill	21.6	19.0	98.0	19.2
Lena	21.6	19.2	99.3	19.5
Peppers	21.4	19.0	99.3	19.3

Table 3.2: Average encoding and decoding time in miliseconds of the proposed and the searchless method for the benchmark images.

All experiments were conducted on an Intel Core 2 Duo E6750 processor, 2.66 GHz with 3GB of RAM running the Linux operating system. The method was implemented in C++ programming language using only integer values and running on a single thread.

The proposed approach is compared to a searchless encoder using the standard grayscale benchmark images *Baboon*, *Barbara*, *Boat*, *Goldhill*, *Lena*, and *Peppers*, with 512 × 512 pixels. The metric used to compare the original and the decoded images is the peak signal-to-noise ratio (PSNR), which can be calculated as:

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{255^2}{\mathrm{MSE}}\right) \tag{3.3}$$

where MSE is the mean squared error between the compared images. The resulting

PSNR for each image using both methods at different bit-rates is shown in Figures 3.6 and 3.7.

The searchless encoder used in the comparisons is based on exactly the same base code from the constant size domain pool encoder, but assigning a fixed value for p_x and p_y of each range block, it is extremely similar, but not exactly equal to the one proposed by Wu et al. [17]. This allows a fair comparison between the two proposed domain-range mappings, avoiding possible differences in the implementation.

Both encoders were compared by varying N_{it} from 100 to 10000 iterations. The rate-distortion curves show that the proposed method maintains a lower distortion in the compressed images at the same rates even though there are fewer range blocks in the transmitted data, since each block uses a larger amount of bits.

Table 3.2 presents the average encoding and decoding times for each image. Although the proposed domain pool is nine times larger if compared to the searchless method, the total encoding time is only increased by a factor of five since there are fewer range blocks. The number of range blocks, as shown in Table 3.3, is smaller because they are subdivided less frequently and consequently they cover larger areas.

Images	Number of Range Blocks	
	searchless	proposed method
Baboon	6005	4223
Barbara	5807	4025
Boat	5708	4025
Goldhill	6005	4223
Lena	5708	4025
Peppers	5708	4124

Table 3.3: Number of range blocks generated at 0.20 bpp for each tested image.

3.7 Conclusions and Future Work

This paper proposed a novel domain pool for fractal coding which is an intermediary between the searchless encoders and the usual methods employing brute force or heuristics in a large domain pool.

Due to the smaller number of encoded range blocks, the proposed method outperforms the searchless encoder even though the latter uses less bits to encode each block. The additional complexity of the larger domain pool has a significant performance impact, however, the encoder still takes approximately 100 ms to encode a 512×512 image compared to several minutes of the original fractal encoders based on large domain pools.

It is important to note that p_x and p_y parameters are transmitted without any form of optimized compression and the domain block is chosen by exhaustive search, which causes a major decrease in the rate-distortion performance and an increase in the encoding time of the proposed approach. Future experiments will analyze possible heuristics to choose the proper domain block and compress its relative position.

Other proposals for future work include the use of a pyramidal algorithm during the image decoding [14], a more efficient method for the quantization and coding of $\bar{r}$ parameter, improvements on the heuristics and data structures used in spatial subdivision, and the use of rate-distortion optimization methods to choose between different domain pools and quantizers [9].

Acknowledgements

The authors are thankful to FAPESP, CNPq and CAPES for the financial support.

3.8 References

[1] Y. Fisher and S. Menlove. Fractal Encoding with HV Partitions. In Y. Fisher, editor, *Fractal Image Compression - Theory and Application*, pages 119–126. Springer-Verlag, London, UK, 1995.

[2] Yuval Fisher. *Fractal Image Compression: Theory and Application.* Springer-Verlag, London, UK, 1995.

[3] S. Furao and O. Hasegawa. A Fast No Search Fractal Image Coding Method. *Signal Processing: Image Communication*, 19(5):393–404, 2004.

[4] Raouf Hamzaoui. Fast Iterative Methods for Fractal Image Compression. *Journal of Mathematical Imaging and Vision*, 11:147–159, October 1999.

[5] A.E. Jacquin. Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations. *IEEE Transactions on Image Processing*, 1(1):18–30, January 1992.

[6] V. Lima, W. R. Schwartz, and H. Pedrini. Fractal Image Encoding Using a Constant Size Domain Pool. In *Workshop de Visão Computacional*, pages 137–142, 2011.

[7] Yong Ho Moon, Hyung Soon Kim, and Jae Ho Kim. A Fast Fractal Decoding Algorithm based on the Selection of an Initial Image. *IEEE Transactions on Image Processing*, 9(5):941–945, May 2000.

[8] G.E. Øien and S. Lepsøy. *A Class of Fractal Image Coders with Fast Decoder Convergence*, chapter Fractal Image Compression, pages 153–175. Springer-Verlag, London, UK, 1995.

[9] A. Ortega and K. Ramchandram. Rate-Distortion Methods for Image and Video Compression. *IEEE Signal Processing Magazine*, 15(6):23–50, 1998.

[10] M. Pi, A. Basu, and M. Mandal. A New Decoding Algorithm based on Range Block Mean and Contrast Scaling. In *International Conference on Image Processing*, volume 2, pages II – 271–4 vol.3, September 2003.

[11] M. Ruhl and H. Hartenstein. Optimal Fractal Coding is NP-Hard. In *Data Compression Conference*, pages 261–270, March 1997.

[12] A. Said. *Lossless Compression Handbook*, chapter Arithmetic Coding. Communications, Networking, and Multimedia. Academic Press, 2003.

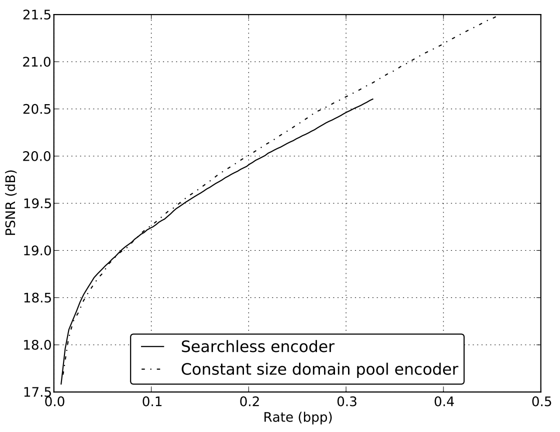
[13] D. Saupe, M. Ruhl, R. Hamzaoui, L. Grandi, and D. Marini. Optimal Hierarchical Partitions for Fractal Image Compression. In *IEEE International Conference on Image Processing*, pages 737–741, Chicago, IL, USA, October 1998.

[14] I. Sutskover and D. Malah. Hierarchical Fast Decoding of Fractal Image Representation using Quadtree Partitioning. In *International Conference on Image Processing and Its Applications*, volume 2, pages 581–585, 1999.

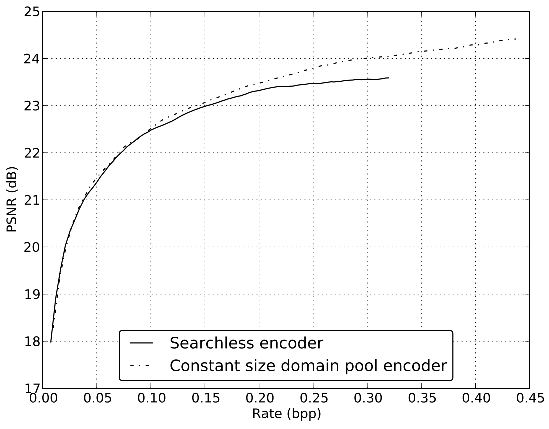
[15] J. Teuhola. Fast Image Compression by Quadtree Prediction. *Real-Time Imaging*, 4:299–308, August 1998.

[16] C.S. Tong and M. Pi. Fast Fractal Image Encoding based on Adaptive Search. *IEEE Transactions on Image Processing*, 10(9):1269–1277, September 2001.

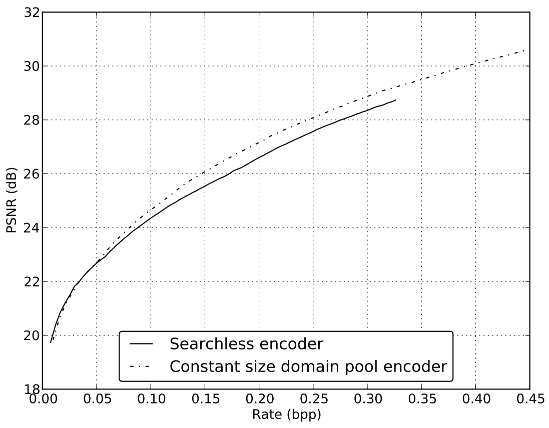
[17] X. Wu, D.J. Jackson, and H. Chen. Novel Fractal Image-Encoding Algorithm Based on a Full-Binary-Tree Searchless Iterated Function System. *Optical Engineering*, 44(10), 2005.



(a) Baboon

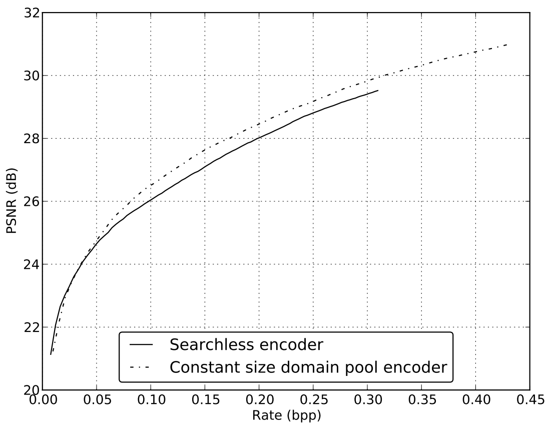


(b) Barbara

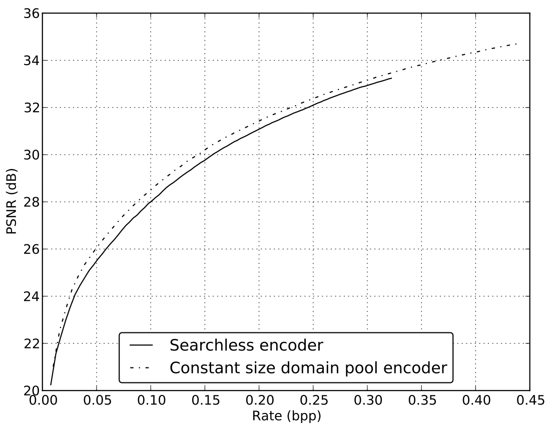


(c) Boat

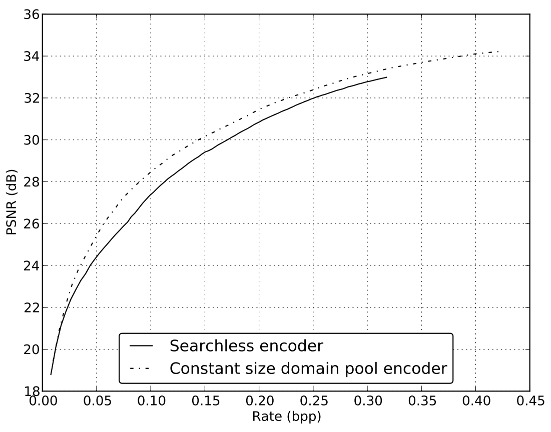
Figure 3.6: Rate-distortion curve for a number of tested images.



(a) Goldhill



(b) Lena



(c) Peppers

Figure 3.7: Rate-distortion curve for a number of tested images.

Capítulo 4

Fast Low Bit-Rate 3D Searchless Fractal Video Encoding

4.1 Prólogo

O artigo [3], apresentado nesta seção, será parte do *24th Conference on Graphics, Patterns and Images (SIBGRAPI'2011)*.

Um método de compressão especializado em cenas com muito movimento comprimidas a taxas de bits por segundo muito baixas é proposto. Grupos de 32 quadros consecutivos são unidos em um fractal volumétrico, que é codificado por meio de uma estrutura espacial capaz de dividir cada bloco molde na horizontal, vertical ou no sentido temporal. Boa parte de sua implementação é extremamente semelhante ao artigo anterior, porém, generalizada para três dimensões e usando apenas um único elemento no *domain pool*.

A abordagem proposta tem um custo computacional reduzido no processo de codificação, pois não usa *Rate-Distortion Optimization* [13].

Em situações com pouco movimento ou com alto *bit-rate*, o codificador x264 possui ampla vantagem, dado que ele não funciona adequadamente em taxas extremas de compressão devido ao seu compensador de movimento que requer uma grande quantidade de dados para representar a diferença entre cada quadro.

Por outro lado, em cenas com grande quantidade de movimento a taxas de compressão extremas, o método proposto possui maior qualidade perceptual.

4.2 Abstract

Video encoding techniques play an important role in data reduction. Fractal compression has received considerable attention in the past decades. While early methods presented prohibitively large encoding times, recent searchless fractal encoders reduced this problem. A fast 3D purely fractal video encoder based on a flexible adaptive spatial subdivision data structure is proposed in this work. The method completely avoids any kind of search for a matching domain block and is capable of performing fast compression and decompression with high visual fidelity. Experimental results show that the developed approach outperforms the state-of-the-art x264 video encoder at very low bit rates in high motion video sequences in both structural dissimilarity measure and encoding time.

4.3 Introduction

Earlier compression methods based on fractal coding [7] suffered from extremely slow encoding times to find an appropriate representation of the image content. Their performance was inferior than more conventional approaches based on invertible transforms, such as the discrete cosine transform (DCT). The main problem is that fractal encoders must find a transform that constructs an approximation of the original image, given the image itself, by looking for similar blocks between different regions using either brute force or a heuristic to reduce the number of elements considered in the search.

Ten years after the introduction of the original fractal encoding method, other methods were proposed by Furao and Hasegawa [5] and Wu et al. [20] aiming to completely avoid any kind of search by imposing a fixed relationship between a block of the original image (called a range block) and its correspondent similar block (called a domain block). This solution produced results comparable to the original JPEG standard and even surpassed it in higher compression ratios while being faster than most of the state-of-the-art encoders.

This paper proposes a fast low bit-rate 3D searchless fractal video compression method for encoding chunks of consecutive frames based on [5, 20]. According to the experimental results, it is perceptually superior to the state-of-the-art x264 [21] at high compression ratios in video sequences with large amount of motion. Additionally, it also presents an encoding time lower than that of x264.

An encoder that can operate at extreme compression ratios is needed by applications that require either low bandwidth communication channels, such as sensor networks and mobile devices, or that must transmit multiple video streams simultaneously, such as surveillance equipments.

The text is organized as follows. Section 4.4 presents a brief review of fractal encoding and related work available in the literature. The proposed method is described in Section 4.5. Experimental results and a comparison between the proposed method and another fractal encoder are presented in Section 4.6. Finally, the conclusions of the work are given in Section 4.7.

4.4 Background

This section initially reviews fractal image and video encoding, including searchless techniques, then describes heuristics for fast rate-distortion and structural dissimilarity measure.

4.4.1 Fractal Image Coding

Unlike other compression algorithms, fractal encoders [7] do not explicitly store an approximation of the image, but they create and transmit a collage, which is a series of instructions that indicate how to partition the image and, for each resulting partitioned region (called a range block), how to generate its content given another block with larger dimensions (called domain block) of the same image. To generate the range blocks, the collage resizes the domain block, applies an affine transform (such as rotation or mirroring), and modifies the gray level values using an equation such as the one proposed by Øien and Lepsøy [12]

$$G(D) = \alpha(D - \bar{D}I) + \bar{r}I \tag{4.1}$$

where G is the gray level transform, D is the downsampled domain block, $\bar{D}$ is the mean value for the domain block, $\bar{r}$ is the mean value for the block in the original scale (the range block), I denotes a matrix filled with ones, and α is the scaling parameter.

The collage is capable of transforming the image into itself given its definition, but it is also capable of transforming any arbitrary signal into an approximation of the original image. In order to decode the image, it is only necessary to apply the collage to any initial image several times until it reaches a fixed point.

The most remarkable characteristics of these methods are the fast image decompression and the extremely slow and complex encoding process, since most approaches construct the collage exhaustively matching each range block with a large number of domain blocks and selecting the best match by certain criteria.

4.4.2 Searchless Fractal Image Coding

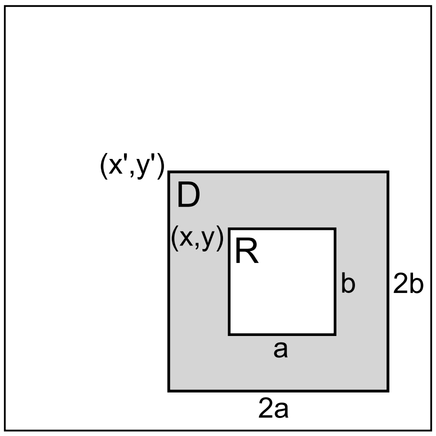


Figure 4.1: Relationship between range (R) and domain (D) blocks in a searchless fractal encoder.

The searchless fractal image coding was first proposed by Furao and Hasegawa [5] to completely avoid searching for the best fit between range and domain blocks. Within this approach for a range block with dimensions a and b located at the coordinate (x, y), there is only one possible domain block with dimensions $2a$ and $2b$ that must be located at (x', y'), as shown in Equation 4.2 and illustrated in Figure 4.1. Each region of the image is encoded only by the α and $\bar{r}$ coefficients of the gray level transform.

$$\begin{aligned} x' &= x - a/2 \\ y' &= y - b/2 \end{aligned} \tag{4.2}$$

The image is initially partitioned into a uniform grid and the collage error (difference between the range block and the transformed domain block) is measured for each region of this partition. If this error is above a certain threshold T_{error}, such region is recursively divided into four subregions and the process continues recursively with the subregions until the block size reaches a minimum size or the collage error drops below T_{error}. Since each block of the image is always recursively divided into another four subregions with equal area, the resulting positions and sizes of each range region can be encoded in a quadtree and α and $\bar{r}$ are transmitted for each region.

A superior method was proposed by Wu et al. [20], which does not impose any limit on the size of the range blocks. To achieve a better compression ratio, there are no scaling parameters for the regions covering one or two pixels and the $\bar{r}$ value is more coarsely quantized in smaller regions. Instead of using a quadtree, this encoder uses a more flexible structure, called binary-tree partition (an example image subdivided by this structure is shown in Figure 4.2), which only divides a region into two sub-blocks with the same size by selecting between splitting it in half in the vertical or horizontal direction. The method presented here uses a volumetric generalization of this data structure.

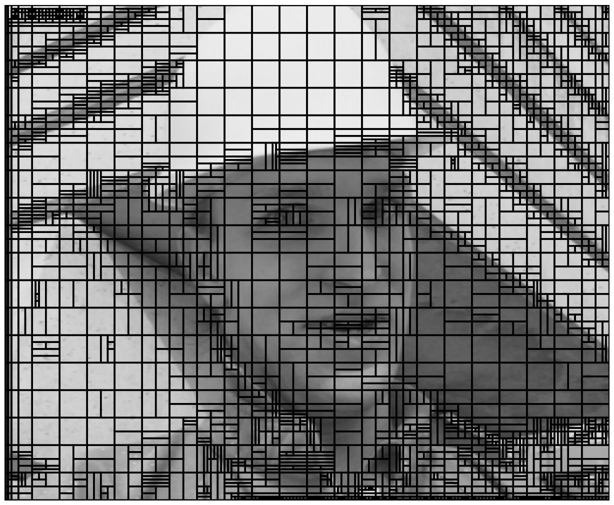


Figure 4.2: An example image subdivided by a recursive binary tree partition.

4.4.3 Fractal Video Coding

The first fractal video encoder was proposed by Hurd et al. [6], which encodes each frame by using blocks from the previous frame as a domain pool either at the same scale as the range blocks or at higher scales, making it a generalization of the usual motion compensation techniques. This approach was later refined by Fisher et al. [4] through the use of quadtrees. More advanced variations of these algorithms were designed later [8, 23].

Another extension of the fractal coding for video sequences was proposed by Lazar and Bruton [10] and Li et al. [11]. In these methods, a chunk of consecutive frames is grouped into a single volumetric image where the x and y axes are the spatial position and the z axis is the time when that pixel value occurred. The collage consists of a spatial subdivision of that volume and, for each resulting range block, the position and the parameters necessary to transform the volumetric domain block. The proposed method is based on this specific generalization of the fractal methods to encode video sequences.

A fast volumetric encoder was later proposed by Chabarchine and Creutzburg [1] for real-time video encoding by simplifying the gray scale transform to use a constant α parameter, resampling every frame to 64×64 pixels, grouping 16 consecutive frames and dividing them into blocks of $16 \times 16 \times 16$ voxels. Each block is represented by an octree and the only possible domain block for each range block is its parent block on the spatial subdivision. The volume is subdivided until a target error threshold is reached. This method is simple and fast, however, its rate-distortion performance is extremely poor.

The volumetric video compression approach [1] was refined by Yao and Wilson [22] through a hybrid method that employs both vector quantization and collages to approximate the original signal. Such hybrid method can encode videos at low bit rates achieving a fair visual quality while being as fast as some MPEG-2 encoders. Unfortunately, this implementation suffers from convergence problems during the decoding stage.

4.4.4 Fast Rate-Distortion Heuristic

Every fractal encoding method must decide how to partition the image into regions that have a similar domain block and the total number of regions must satisfy the restriction on the number of bits set by the user. Most encoders (such as the one described in Section 4.4.2) subdivide each region recursively until a certain threshold for the collage error is reached. This heuristic tries to guarantee a minimum reconstruction quality for the resulting decoded image, but it is difficult to efficiently satisfy any restriction on the total size of the collage.

A solution to this problem was proposed by Saupe et al. [15] after investigating optimal partitions in fractal encoding. In this method, the image is divided into a uniform grid, where each region has its collage error calculated and inserted into a priority queue. At each iteration, the region with the highest error is removed from the queue and subdivided, then its subregions are inserted into the queue.

The size of the collage increases slightly at each iteration, so it is possible to achieve a certain size by stopping the heuristic after a certain number of iterations, resulting in an approximation of the desired size. The heuristic is also intuitive since the most distorted regions of the image have priority over the other ones.

4.4.5 Structural Dissimilarity

Most comparisons between video and image encoders are based on metrics derived from the sum of squared differences (SSD) or the mean squared error (MSE). The SSD and the MSE between two images A and B with size $W \times H$ is given by

$$\mathrm{SSD}(A,B) = \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} (A_{x,y} - B_{x,y})^2 \tag{4.3}$$

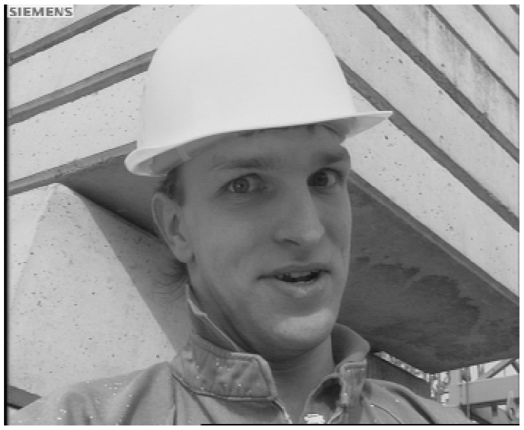
$$\mathrm{MSE}(A,B) = \frac{\mathrm{SSD}(A,B)}{W \times H} \tag{4.4}$$

A critical issue with the MSE is that it does not measure the resulting image quality directly and it can attribute similar scores to images with large differences in psychovisual quality. As illustrated in Figure 4.3, the psychovisual quality degradation between the images is measured by the SSIM, while the MSE does not reflect that fact, as indicated in the figure captions.

The structural similarity index (SSIM) [18] was proposed as a metric to compare images which correlates more appropriately with the human perception. It maps two images into an index in the interval $[-1, 1]$, where higher values are given to more

(a) Original image (SSIM=1.0, MSE=0)

(b) Multiplied by 1.072 (SSIM=0.995837, MSE=145.96)


(c) Subtracted by 12 pixels (SSIM=0.994981, MSE=143.97)

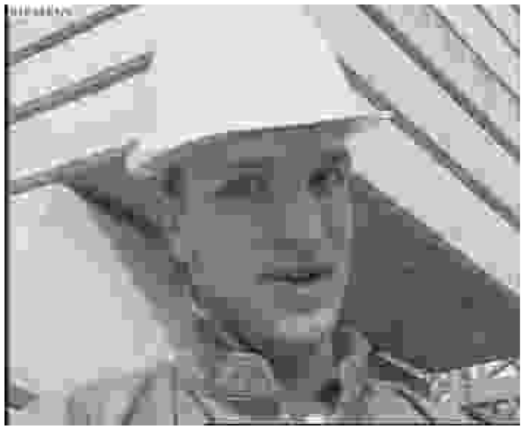

(d) Compressed by JPEG (SSIM=0.742805, MSE=142.91)

Figure 4.3: Several distorted versions of the same image with different perceptual qualities and approximately the same MSE.

similar pairs of images, calculated as

$$\mathrm{SSIM}(A, B) = \frac{(2\mu_A\mu_B + c_1)(2\sigma_{AB} + c_2)}{(\mu_A^2 + \mu_B^2 + c_1)(\sigma_A^2 + \sigma_B^2 + c_2)} \tag{4.5}$$

where μ_A, μ_B, σ_A^2 and σ_B^2 are the averages and variances of A and B, σ_{AB} is the covariance between A and B, and both c_1 and c_2 are predefined constants. This metric is calculated as the average of the score between several blocks using a sliding window of 11×11 pixels.

The structural similarity scores are also shown in Figure 4.3. In this example, it is possible to notice that the image with the lowest MSE is the least similar to the original. In addition, all three images have almost the same MSE, but the structural similarity is more coherent to what one would expect from a comparison metric.

The structural dissimilarity is a derived metric from the structural similarity that results in more distinct values, since a small variation in the original SSIM indicates a large difference in image quality. It is given by

$$\mathrm{DSSIM}(x, y) = \frac{1}{1 - \mathrm{SSIM}(x, y)} \tag{4.6}$$

4.5 Proposed Method

The proposed 3D video encoder constructs a volumetric image composed of 32 consecutive frames and transmits a collage that is used to reconstruct them (using the same definition as the volumetric encoders described in Section 4.4.3). This image is divided into a uniform grid of blocks with $16 \times 16 \times 16$ voxels and each one of these blocks has its own spatial subdivision tree. In this binary tree, each block has two subblocks with equal volume which are created splitting their parent into half in the horizontal, vertical or temporal direction. The blocks are subdivided according to the heuristic presented in Section 4.4.4, using the SSD of the collage error as the distortion metric.

The SSD was chosen as the distortion metric since the MSE disregards the size of the block, giving the same score to equally distorted blocks with large differences in volume. The SSIM was not created to evaluate volumetric images, it must be calculated using sliding windows since it cannot properly compare two isolated subblocks of an image and, contrary to other usual metrics, it is impossible to estimate the final SSIM of the image given the SSIM of each range block. In the case of using SSD, the final score is the sum of the score of all the encoded blocks. Given these observations, if the SSIM was employed the heuristic would not estimate which block can result in the largest reduction of distortion, since the distortion metric itself cannot be efficiently measured in either the entire image or the range blocks.

This heuristic requires a given volumetric block to be encoded and split in case it is chosen during an iteration. The block encoding method uses a relationship between range and domain blocks similar to the one used in bidimensional searchless fractal encoders, but generalized to three dimensions. For each range block with dimensions a, b and c located at (x, y, z), there is only one possible domain block that can be used to represent it. The position of the domain block can be expressed in Equation 4.7, with dimensions equal to $2a$, $2b$ and $2c$. This block is illustrated in Figure 4.4.

$$\begin{aligned} x' &= x - a/2 \\ y' &= y - b/2 \\ z' &= z - c/2 \end{aligned} \tag{4.7}$$

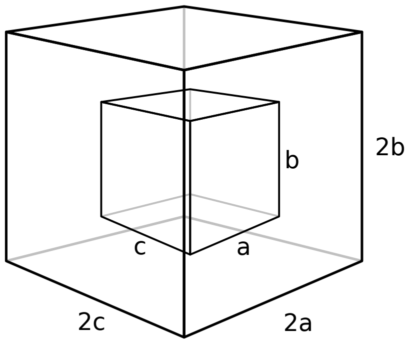


Figure 4.4: An example of the relationship between range and domain blocks in the proposed 3D encoder.

Some blocks cannot use this equation since applying it would result in a domain block that is not completely inside the volumetric image. In these cases, if a coordinate is negative, it is set to zero and if any pixel inside the block has a coordinate larger than the dimensions of the image, the coordinate of the domain block is set to be the dimension of the image minus the dimension of the block.

Each range block is matched to its only respective domain block using Equation 4.1 by assigning the α parameter as 0.25, 0.5, 0.75 or 1.0, as suggested by Tong and Pi [16].

The heuristic used to decide how to properly split a block divides it into all the three possible directions and the direction resulting in the smallest sum of the SSD for both resulting subblocks is chosen.

All the required symbols and parameters are encoded using a context-adaptive arithmetic coder [14]. Each range block is encoded by its α parameter, which occupies

Decision Table for the Quantization of $\bar{r}$		
Volume	Quantization step	Number of used bits
1	16	4
2	16	4
4	16	4
8	8	5
16	8	6
32	4	6
64	4	6
128	2	7
256	2	7
≥ 512	1	8

Table 4.1: Quantizers applied according to the volume of the range block.

2 bits in the worst case, along with $\bar{r}$, which is quantized according to the range block volume as shown in Table 4.1. For range blocks with one or more dimensions smaller than 2 pixels, the only transmitted parameter is $\bar{r}$. Along with these parameters, the spatial subdivision tree for each block in the initial uniform subdivision is coded by a sequence of symbols pointing to the decoder, in a depth-first order, whether a certain region was subdivided or not and in which direction it was split. The α parameter and the binary decision symbols in the spatial subdivision tree have their own high order adaptive contexts, one for each possible value of $\lfloor \log V \rfloor$, where V is the total volume of the encoded block. The direction which each block is split is encoded by another set of 3 high order adaptive contexts chosen according to the direction used to split its parent.

The $\bar{r}$ parameter is encoded as a difference between a quantized prediction calculated as the average of $\bar{r}$ of the neighboring blocks located at the top, to the left and behind the encoded block and the real quantized value. This difference is encoded by the Adaptive Goulomb-Rice code described in [19], using one context for each possible $\lfloor \log V \rfloor$ in the same manner as the other parameters.

4.6 Experimental Results

All experiments were conducted on an Intel Core 2 Duo E6750 processor, 2.66 GHz with 3GB of RAM running Linux operating system. The method was implemented using the C++ programming language without any additional optimizations.

The proposed approach is compared to the state-of-the-art H.264 encoder, called x264 [21], using the standard grayscale benchmark sequences 'Foreman', 'Car phone', 'Bus', 'Football', 'Akiyo', 'Miss America', 'Bowing', and 'Hall monitor' in the CIF format [2]. The length of each sequence is shown in Table 4.2.

Sequence	# Frames
Foreman	300
Carphone	457
Bus	150
Football	260
Akiyo	300
Hall monitor	300
Bowing	300
Miss America	179

Table 4.2: Number of frames for the used video sequences.

The x264 was configured to closely match the behavior of the proposed encoder by forcing it to insert a keyframe at every 32 frames and compiling it without any CPU specific optimizations. The command line used to invoke this encoder was

```
x264 --tune ssim --preset medium --profile baseline
--keyint 32 --bitrate 'target bitrate'
```

The video sequences in the following experiments were encoded at low bit rates, which implies that the results had high distortions when measured by Equation 4.3. As shown in [18], the ambiguity of the metrics derived from the SSD from a perceptual point of view is high and becomes even larger as the distortion increases. It is important to observe that both α and $\bar{r}$ are quantized (i.e. they must assume one of a small set of possible values instead of being continuous) which causes a mean shift and a contrast change in every range block, even though the effect of these quantizations is perceptually negligible. In order to ensure a proper comparison between both methods, the mean structural dissimilarity for both encoders is used in the experiments. This metric is widely accepted for its simplicity and reasonably

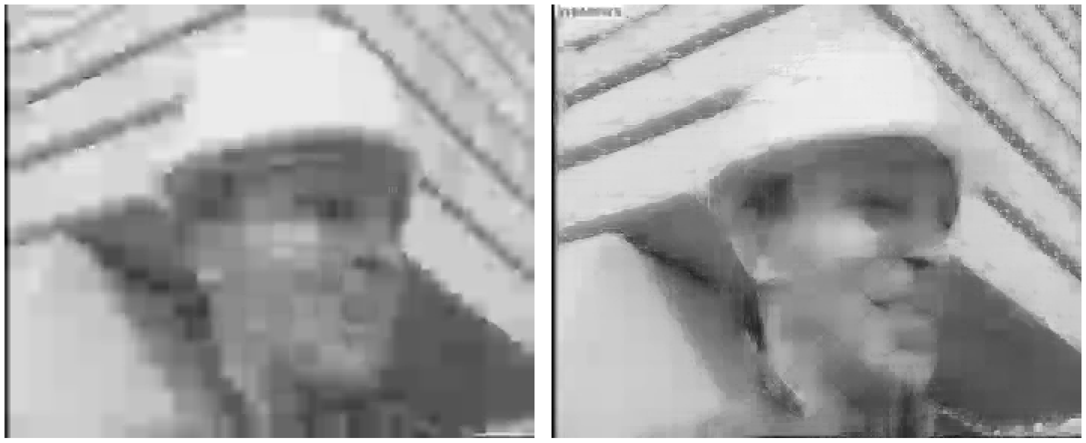

(a) A frame from the 'Foreman' sequence encoded by x264 (b) The same frame encoded by the proposed method

Figure 4.5: Differences in the accuracy of frame prediction in both methods at 50 kbps.

accuracy, being employed in the design of several image encoders, such as [9], and has a built-in implementation in the x264.

The bit rate was varied to closely match the same values in both encoders. As observed in Figure 4.6, the proposed encoder outperforms the x264 codec at very low bit rates in these high motion sequences. This is due to the motion compensation algorithm of the H.264 encoder cannot operate properly in these conditions given that an accurate prediction of each frame would require a large amount of bits. An example of this case is shown in Figure 4.5, where the motion of smooth regions is ignored or poorly represented by the x264, then generating temporal artifacts. In this sequence, the proposed method accurately represents most of the moving regions, causing high distortions only on instantaneous movements such as eye blinking. The 'Car phone' and 'Foreman' sequences have transitions between high and low motion scenes giving an advantage to x264 at bit rates larger than 60 kbps.

The Figure 4.7, which are related to scenes with a static background and one or two moving objects, show that in these cases the highly efficient transform coding of the x264 has a significant advantage over the proposed fractal encoder and the motion compensation is achievable due to the localized motion of a few regions in each frame.

The total encoding time of the proposed method is remarkably low as evidenced in Figures 4.8 and 4.9. Both methods, the proposed fractal encoder and the x264 codec, were implemented in high level languages without excessive optimizations to ensure a fair comparison between them. The proposed method takes between one third and one fifth of the total time needed by the x264 to encode the same content.

4.7 Conclusions and Future Work

This paper proposed a 3D searchless fractal video encoder that is comparable to the x264 encoder [21] at very low bit rates. As it has been observed from the experimental results, while the proposed encoder outperformed the x264 in the high motion video sequences, for scenes with a static background and few moving objects, the x264 presented advantage over the proposed method. Furthermore, contrary to most fractal-based methods, it presents a very low encoding time even when compared to x264 for all tested sequences.

Suggestions for future enhancements in the proposed approach include a better lossless encoding of the gray level parameters and of the symbols used in the spatial subdivision, the use of fractal interpolation to encode the content at a lower frame rate and super-sample it to the original rate, the implementation of more complex gray level transforms such as the one used in [17] and, finally, the use of rate-distortion optimization methods [13] to choose which quantizers to use and which regions must be split.

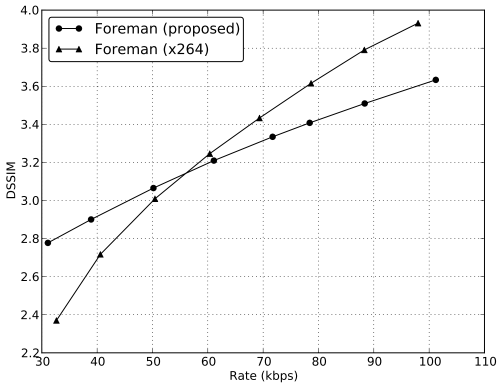
Acknowledgments

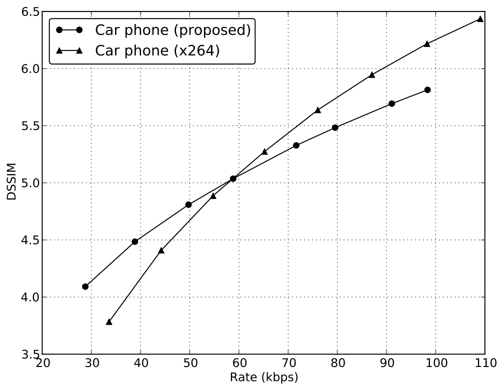
The authors are thankful to FAPESP, CNPq and CAPES for the financial support.

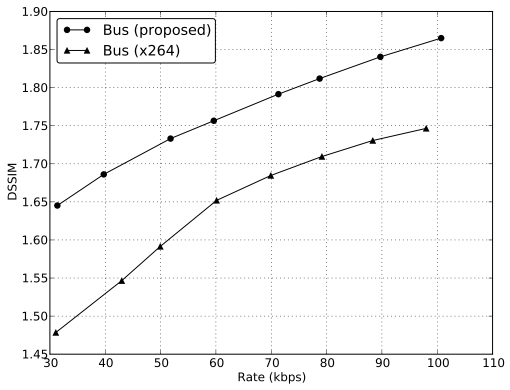
4.8 References

[1] A. Chabarchine and R. Creutzburg. 3D Fractal Compression for Real-Time Video. In *2nd International Symposium on Image and Signal Processing and Analysis*, pages 570–573, 2001.

[2] CIPR. Sequences. `http://www.cipr.rpi.edu/resource/sequences/`.







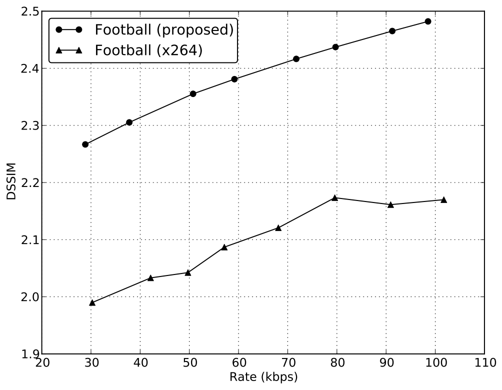


Figure 4.6: Mean structural dissimilarity at different rates for the proposed video encoder and the x264 encoder.

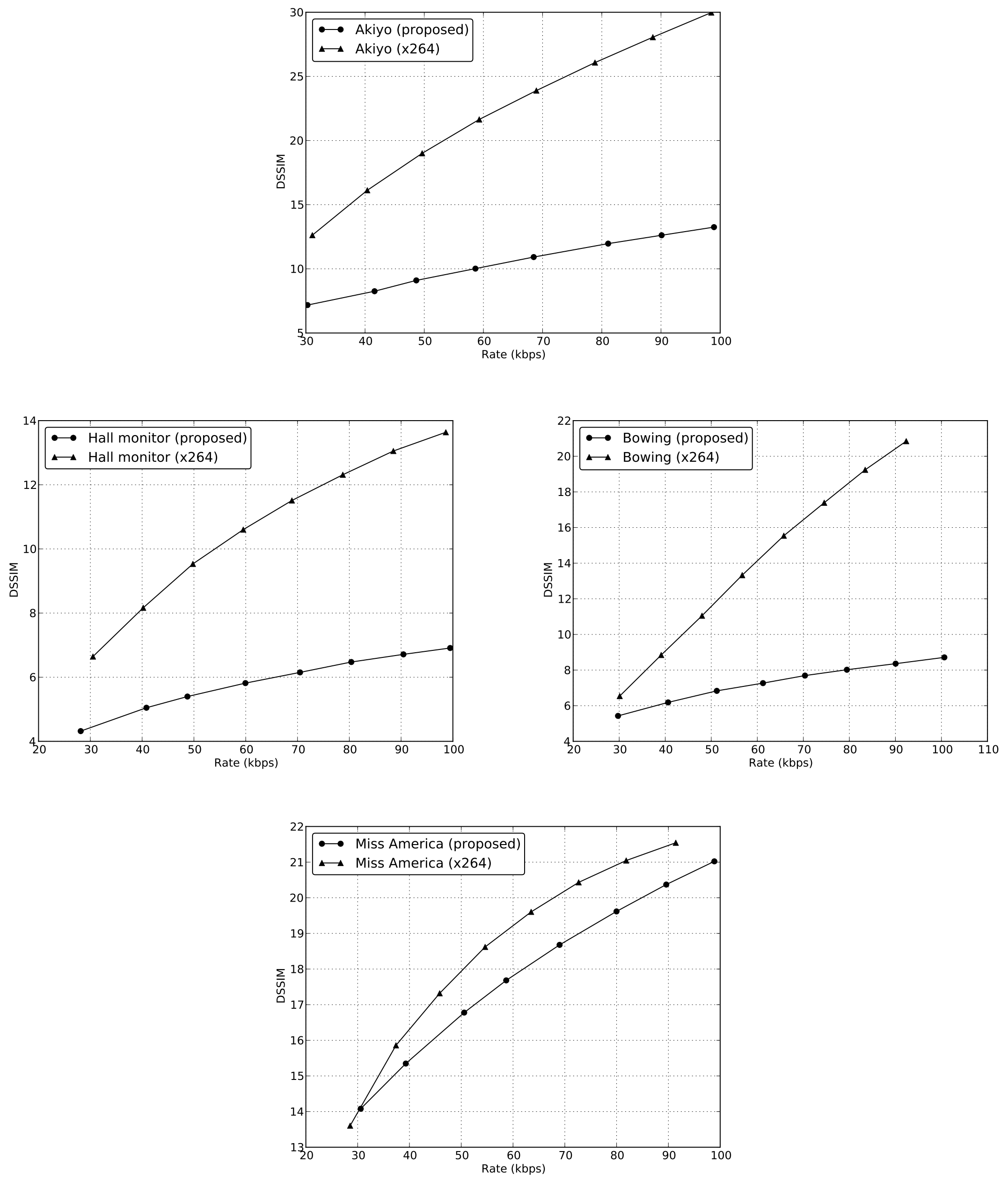


Figure 4.7: Mean structural dissimilarity at different rates for the proposed video encoder and the x264 encoder.

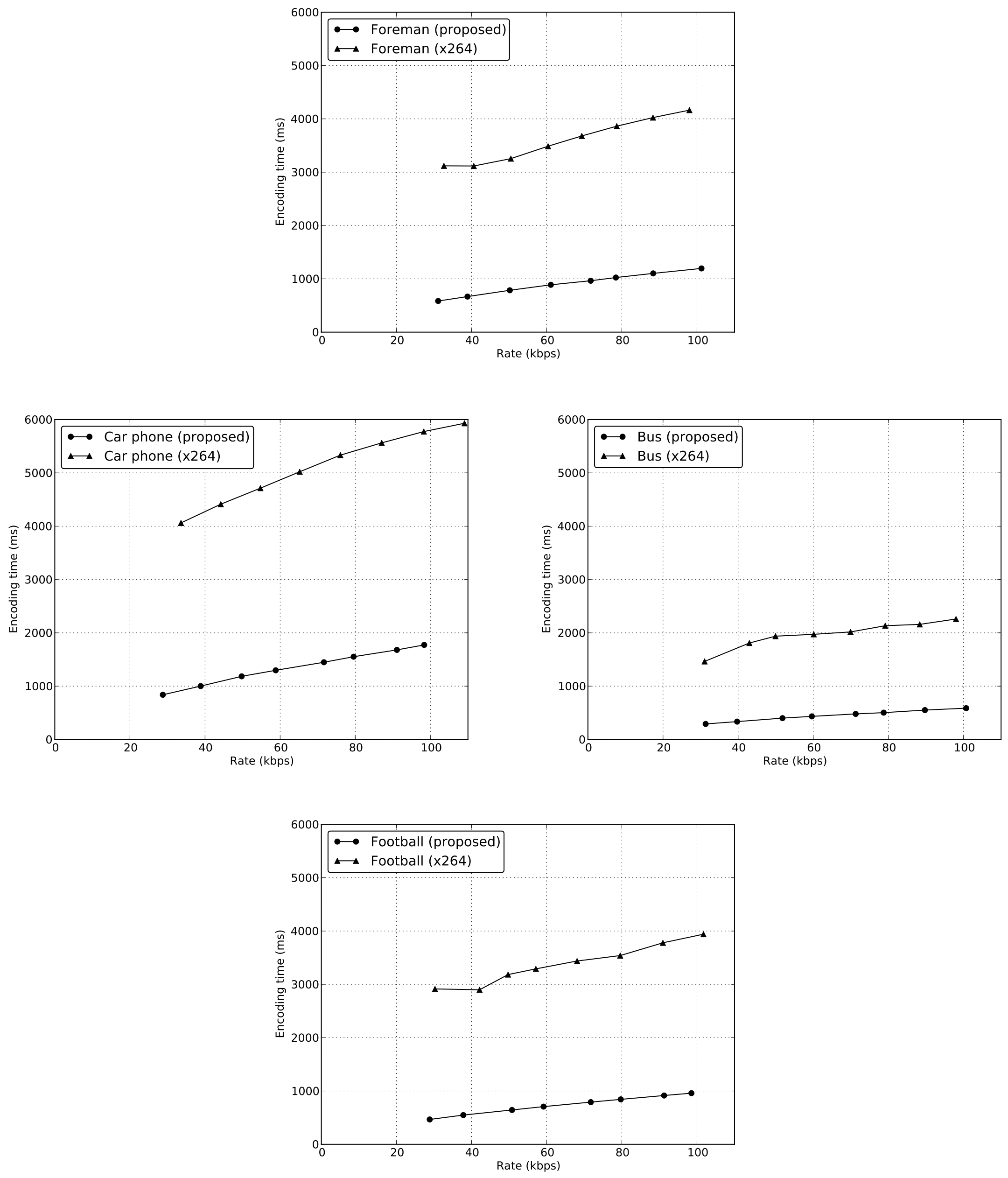


Figure 4.8: Mean structural dissimilarity at different rates for the proposed video encoder and the x264 encoder.

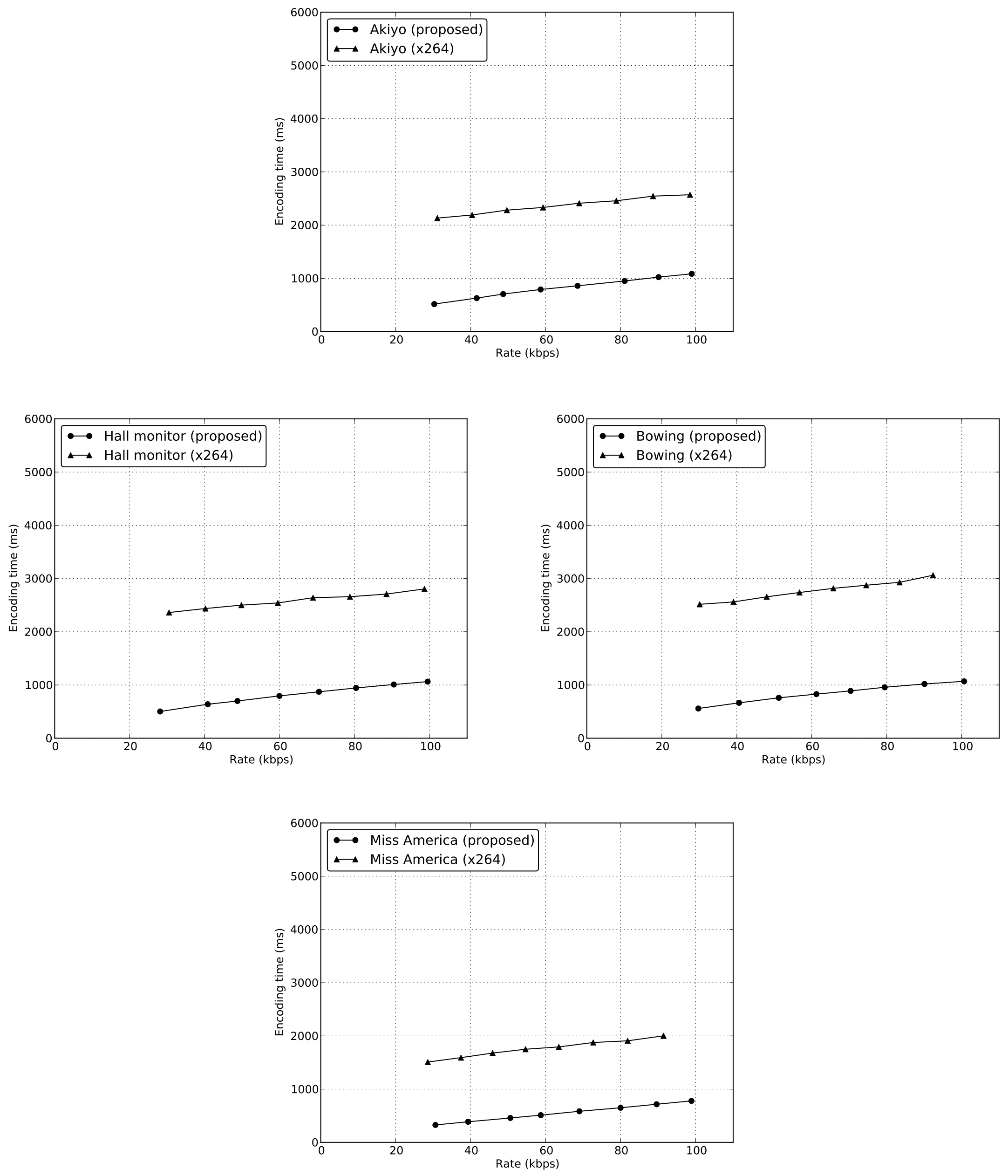


Figure 4.9: Encoding time at different rates for the proposed video encoder and the x264 encoder.

[3] V. de Lima, W. R. Schwartz, and H. Pedrini. Fast Low Bit-Rate 3D Searchless Fractal Video Encoding. In *Conference on Graphics, Patterns and Images*, Maceio, AL, Brazil, 2011.

[4] Y. Fisher, D.N. Rogovin, and T.J. Shen. Fractal (Self-VQ) Encoding of Video Sequences. *Visual Communications and Image Processing*, 2308(1):1359–1370, 1994.

[5] S. Furao and O. Hasegawa. A Fast No Search Fractal Image Coding Method. *Signal Processing: Image Communication*, 19(5):393–404, 2004.

[6] L.P. Hurd, M.A. Gustavus, and M.F. Barnsley. Fractal Video Compression. In *Thirty-Seventh IEEE Computer Society International Conference*, pages 41–42, February 1992.

[7] A.E. Jacquin. Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations. *IEEE Transactions on Image Processing*, 1(1):18–30, January 1992.

[8] C. Kim, R. Kim, and S. Lee. Fractal Coding of Video Sequence using Circular Prediction Mapping and Noncontractive Interframe Mapping. *IEEE Transactions on Image Processing*, 7(4):601–605, April 1998.

[9] P. K. Krause. FTC - Floating Precision Texture Compression. *Computers and Graphics*, 34:594–601, October 2010.

[10] M.S. Lazar and L.T. Bruton. Fractal Block Coding of Digital Video. *IEEE Transactions on Circuits and Systems for Video Technology*, 4(3):297–308, June 1994.

[11] H. Li, M. Novak, and R. Forchheimer. Fractal-Based Image Sequence Compression Scheme. *Optical Engineering*, 32(7):1588–1595, 1993.

[12] G.E. Øien and S. Lepsøy. *A Class of Fractal Image Coders with Fast Decoder Convergence*, chapter Fractal Image Compression, pages 153–175. Springer-Verlag, London, UK, 1995.

[13] A. Ortega and K. Ramchandram. Rate-Distortion Methods for Image and Video Compression. *IEEE Signal Processing Magazine*, 15(6):23–50, 1998.

[14] A. Said. *Lossless Compression Handbook*, chapter Arithmetic Coding. Communications, Networking, and Multimedia. Academic Press, 2003.

[15] D. Saupe, M. Ruhl, R. Hamzaoui, L. Grandi, and D. Marini. Optimal Hierarchical Partitions for Fractal Image Compression. In *IEEE International Conference on Image Processing*, pages 737–741, Chicago, IL, USA, October 1998.

[16] C.S. Tong and M. Pi. Fast Fractal Image Encoding based on Adaptive Search. *IEEE Transactions on Image Processing*, 10(9):1269–1277, September 2001.

[17] X. Wang, Y. Wang, and J. Yun. An Improved No-search Fractal Image Coding Method Based on a Fitting Plane. *Image and Vision Computing*, 28:1303–1308, August 2010.

[18] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600 –612, April 2004.

[19] M.J. Weinberger, G. Seroussi, and G. Sapiro. LOCO-I: A Low Complexity, Context-based, Lossless Image Compression Algorithm. *Data Compression Conference*, pages 140–149, 1996.

[20] X. Wu, D.J. Jackson, and H. Chen. Novel Fractal Image-Encoding Algorithm Based on a Full-Binary-Tree Searchless Iterated Function System. *Optical Engineering*, 44(10), 2005.

[21] x264. Video Encoder. `http://www.videolan.org/developers/x264.html`.

[22] Z. Yao and R. Wilson. Hybrid 3D Fractal Coding with Neighbourhood Vector Quantisation. *EURASIP Journal on Applied Signal Processing*, 2004:2571–2579, January 2004.

[23] Shiping Zhu, Zaikuo Wang, and K. Belloulata. A Novel Fractal Monocular and Stereo Video Codec based on MCP and DCP. In *IEEE International Conference on Industrial Technology*, pages 168–172, March 2010.

Capítulo 5

Fast Adaptive Transcoding Based on a 3D Low Bit Rate Fractal Video Encoder

5.1 Prólogo

O artigo [15], apresentado nesta seção, foi submetido ao periódico *IEEE Transactions on Multimedia.*

Baseado no método de compressão do artigo anterior, uma adaptação simples da heurística descrita na Seção 1.3 permitiu que o vídeo fosse pré-processado de tal forma a poder ser traduzido em tempo real para várias taxas-alvos de bits por segundo. Outros métodos semelhantes não conseguem ser rápidos na tradução ou fornecem uma quantidade pequena de taxas de bits por segundo possíveis.

Outras mudanças incluem o uso do método de construção de colagems proposto no artigo *Fractal Image Encoding Using a Constant Size Domain Pool*, desta vez generalizado para vídeos por meio de um *domain pool* de 27 elementos. Os blocos moldes são divididos pela metade por um plano de corte que minimiza, para ambos os sub-blocos, o produto entre o volume e a variância destes.

O preditor das médias dos blocos molde usa como predição a média ponderada de blocos vizinhos já transmitidos, sendo que o peso é a área de contato com o bloco sendo transmitido.

5.2 Abstract

Video transmissions usually occur at a fixed or at a small number of predefined bit rates. This can lead to several problems in communication channels whose bandwidth can vary along time (e.g. wireless devices). This work proposes a video encoding method for solving such problems through a fine rate control that can be dynamically adjusted with low overhead. The encoder uses fractal compression and a simple rate distortion heuristic to preprocess the content in order to speed up the process of switching between different bit rates. Experimental results show that the proposed approach can accurately transcode a preprocessed video sequence into a large range of bit rates with a small computational overhead.

5.3 Introduction

Most video streaming services use a reliable point-to-point channel to transmit videos and usually the bit rate is fixed or can be changed only to a few different possibilities, which might cause visual interruptions during the transmissions if the available bandwidth of the channel is variable. This behavior frequently occurs in wireless communication. Therefore, if the video server could adapt itself to the client's bandwidth, the user would have both the best possible quality given the available bandwidth and the least amount of interruptions.

A proposed solution to this problem is called *transcoding*, which converts the video stream into another one satisfying a given constraint. Most of the proposed methods [9] are extensions to well-known DCT-based video encoders and are capable of changing the frame rate, bit rate, spatial resolution or the standard used in the transmission. Another approach is Scalable Coding [23], based on transmitting a single stream divided into layers that can be acquired separately according to the available bandwidth.

This work proposes an approach that compresses the video at the maximum desired transmission rate and includes some extra data. This data is used to transcode the compressed video to a large range of bit rates. The target bit rate of this process can be dynamically adjusted with low overhead and the transcoding algorithm is near optimal in a sense that it must only read the compressed file, execute a binary search in a table to find the correct operating parameters and write the resulting transcoded file.

To the best of our knowledge, the proposed approach is the first transcoding method based on fractal video encoding. It relies exclusively on changing the resulting bit rate, taking advantage of the spatio-temporal independency of the fractal codes to avoid any changes to the frame rate or the spatial resolution.

In order to reduce the complexity of the algorithm, the encoding is extremely simplified when compared to other fractal-based approaches while still maintaining an acceptable rate-distortion performance. Perceptual quality comparisons with the x264 state-of-the-art encoder are presented.

This paper is organized as follows. The video encoder, described in Section 5.5, is based on volumetric and searchless fractal methods reviewed in Section 5.4. Experimental results with well-known video sequences are presented in Section 5.6 and, finally, the conclusions and future work appear in Section 5.7.

5.4 Background

This section presents a brief review of fractal image encoding (Section 5.4.1), the searchless method for constructing collages (Section 5.4.2), a restricted domain pool proposed earlier[16] (Section 5.4.3), some related fractal video encoders available in the literature (Section 5.4.4), a searchless fractal video encoding method (Section 5.4.5), a heuristic to control the encoding process (Section 5.4.6) and a perceptual quality metric used in image comparisons (Section 5.4.7).

5.4.1 Fractal Image Encoding

Fractal image encoders transmit a fractal that approximates the original image. The first of such methods was proposed in the seminal paper by Jacquin [10], where the method creates and transmits an operator, called collage, capable of reconstructing an approximation of the original image given a subsampled version of it. During the decoding process, the collage is applied to an arbitrary initial image, the result is subsampled and this process is repeated until the image converges to a fixed point. At first glance, this could imply that the subsampled image must be transmitted, however, the collage is constructed in such way that when applied several times to any image with the correct dimensions, it converges to almost the same approximation that would be achieved if it was applied to the original subsampled image.

The usual process used to construct the collage partitions the image into blocks (called range blocks) and each one of them is matched with a same-sized block in the subsampled image (called domain block) after being transformed by a prespecified function. This match is done either by exhaustive search or through the use of specialized heuristics. In general, the matching process is very time consuming; therefore, most fractal methods have extremely slow and complex encoding processes, but the decoding is usually significantly faster.

The collage can rotate, flip or mirror the domain blocks and apply a transform into ther gray level values, such as the one used by Tong and Pi [24]

$$G(D) = \alpha(D - \bar{D}J) + \bar{r}J \tag{5.1}$$

where G is the gray level transform, D is the downsampled domain block, $\bar{D}$ is the mean value for the domain block, $\bar{r}$ is the mean value for the block in the original scale (the range block), J denotes the unit matrix, and α is a scale parameter.

5.4.2 Searchless Fractal Encoding

The searchless fractal image coding was introduced by Furao and Hasegawa [5] as a less complex alternative algorithm for constructing collages. In this approach, each range block has only one candidate domain block to be matched. If this matching does not achieve the desired reconstruction quality, the range block is divided into four blocks (which can be seen as a tree-based decomposition) and the process continues recursively until the range blocks reach a certain minimum size.

The only candidate domain block to be matched against a range block with dimensions a and b and located at (x, y) has the following coordinates

$$\begin{aligned} x' &= x - a/2 \\ y' &= y - b/2 \end{aligned} \tag{5.2}$$

where its dimensions are $2a$ and $2b$. This relationship is shown in Figure 5.1. In order to simplify the encoding process, the domain blocks are transformed by Equation 5.1, without any additional processing. Therefore, each range block can be encoded only by its α and $\bar{r}$ coefficients.

This approach was refined by Wu et al. [27] by dividing each range block in half either in the vertical or horizontal direction and without imposing any limits to the size of the range blocks. The smaller blocks have their $\bar{r}$ parameter more coarsely quantized than the larger ones, and blocks with only one or two pixels are forced to have their α equal to zero.

5.4.3 Domain Pool with Constant Size

A modification proposed to the fast searchless encoders described in Section 5.4.2 was presented by Lima et al. [16]. Instead of matching each range block to a single domain block, one of nine different range-domain mappings is chosen. For a range block with dimensions $a \times b$ located at (x, y), the blocks in its domain pool have dimensions equal to $2a \times 2b$ and their positions are given by

$$\begin{aligned} x' &= x - a + p_x \times a/2 \\ y' &= y - b + p_y \times b/2 \end{aligned} \tag{5.3}$$

where both p_x and p_y must be equal to 0, 1 or 2. All possible domain blocks given by this equation are illustrated in Figure 5.1.

5.4.4 Fractal Video Encoding

The first fractal video encoder was proposed by Hurd et al. [8] by creating a collage that transforms the previous frame into the next one. This transform could use either blocks from the original scale or from a subsampled version of the frame. This method is very similar to the motion compensation usually found in most video encoders. This approach was enhanced by Fisher et al. [4] by varying the size of each used range block through a quadtree. Refined methods based on these concepts were proposed by Kim et al. [11] and Zhu et al. [30].

The fractal video encoding method proposed by Lazar and Bruton [13] and Li et al. [14] used tridimensional collages that transform a subsampled version of a volume formed by consecutive frames into the original signal by matching volumetric range and domain blocks. Due to the extra dimension, this causes the encoding process to be even more time consuming when compared to the image encoders.

A faster variation of this volumetric approach was proposed by Chabarchine and Creutzburg [1] by using a simpler gray scale transform, an extremely restricted domain pool for each range block and a simple spatial subdivision structure. This

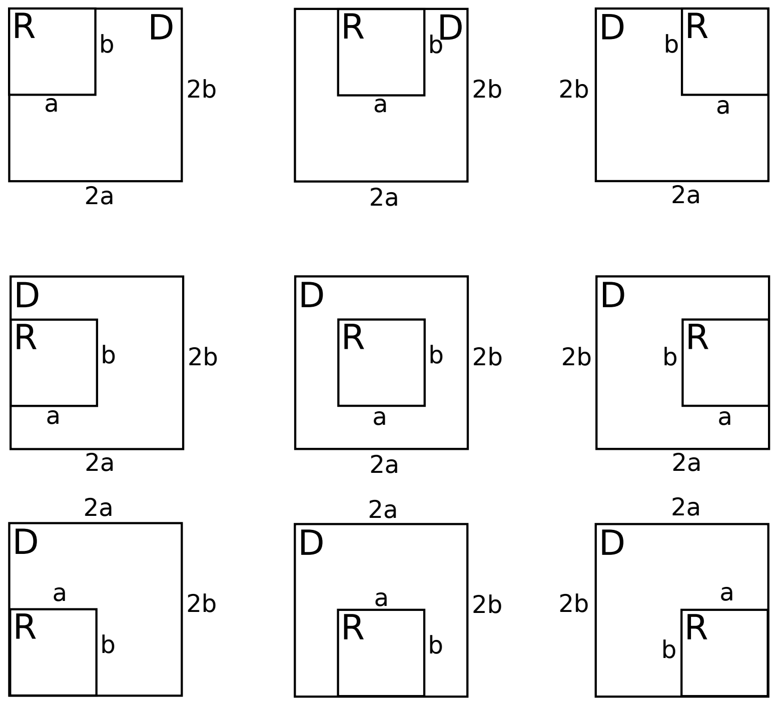


Figure 5.1: All possible relationships between the range (R) and the domain blocks (D) in the encoder proposed by Lima et al. [16].

method was capable of encoding videos at real-time, however, its rate-distortion performance was poor.

Another refined volumetric fractal encoder was introduced by Yao and Wilson [29] by using both vector quantization and domain blocks to approximate the signal. The approach could achieve a fair visual quality at low bitrates while being relatively fast, but its decoder suffered from convergence problems.

5.4.5 Searchless Fractal Video Encoding

A fast method for low bit rate video compression was introduced by Lima et al. [3], where its perceptual performance was comparable to the state-of-the-art, while having a lower computational complexity. It is a generalization of the algorithms given in Section 5.4.2 which encodes a group of consecutive frames as a volumetric fractal.

This group is divided into a uniform grid of tridimensional range blocks with $16 \times 16 \times 16$, each one of them matched against a single domain block. The algorithm closely resembles bidimensional approaches, such that if a certain range block is poorly represented, it is divided in half in any direction (temporal, horizontal or vertical) and the resulting subblocks are either recursively divided or encoded.

In this method, a range block with dimensions a, b and c at the position x, y and z is matched with a domain block with dimensions equal to $2a$, $2b$ and $2c$ located at

$$\begin{aligned} x' &= x - a/2 \\ y' &= y - b/2 \\ z' &= z - c/2 \end{aligned} \tag{5.4}$$

such that this relationship is illustrated in Figure 5.2. The heuristic used to control the rate and distortion of the encoded video is presented in Section 5.4.6. The decoding process, similar to the one presented in Section 5.4.1, applies a collage between a subsampled version of the group of frames to construct the approximation at the original resolution.

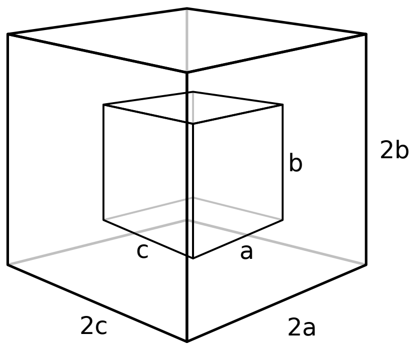


Figure 5.2: The relationship between the range (R) and the domain block (D) used in the searchless fractal video encoder.

5.4.6 Rate-Distortion Heuristic for Fractal Encoding

Saupe et al. [22] proposed a low-complexity heuristic for partitioning an image to obtain simultaneously a proper rate distortion performance and an accurate rate control.

The heuristic initially divides the image into a uniform partition of blocks with the same size and inserts them into a priority queue which sorts the blocks according to their mean squared error (MSE). At each iteration, the heuristic takes the block with the highest MSE, divides it into two sub-blocks which are properly encoded and inserted again into the queue Q. This heuristic is intuitive since the most poorly represented regions of the image have priority over others and the size of the encoded data increases slightly at each step allowing a fine rate control.

Algorithm 3 The Rate-Distortion heuristic proposed by Saupe et al. [22]

```
1: Subdivide the image into an initial uniform grid of range blocks
2: Insert the resulting blocks in the priority queue $Q$ sorting them through their MSE
3: $i = 0$
4: repeat
5:    Remove the block with the largest MSE from $Q$
6:    Subdivide it into two new blocks
7:    Insert these new blocks into $Q$
8:    $i = i + 1$
9: until The desired size is reached or $i < i_{upper}$
```

This algorithm has three critical properties. It is incremental, therefore the rate control is achieved by stopping the algorithm at a certain iteration that generates a compressed data with the desired size or at a certain constant number of iterations i_{upper} (more complex optimization methods for rate control require the content to be encoded using several schemes and just one of them is selected). The heuristic does not prune the tree and spends computational effort only on the range blocks which are subdivided. Finally, to reach a certain rate constraint, the heuristic passes through a set of solutions with a wide range of different target rates until it stops.

5.4.7 Structural Dissimilarity

Most comparisons between video and image encoders are based on metrics derived from the sum of squared differences (SSD) or the mean squared error (MSE). The

SSD and the MSE between two images A and B with size $W \times H$ are given by

$$\mathrm{SSD}(A,B) = \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} (A_{x,y} - B_{x,y})^2 \tag{5.5}$$

where $A_{x,y}$ and $B_{x,y}$ are the intensity of a pixel located at (x, y) in A and B, respectively.

$$\mathrm{MSE}(A,B) = \frac{\mathrm{SSD}(A,B)}{W \times H} \tag{5.6}$$

A critical issue with the MSE is that it does not measure the resulting image quality directly and it can attribute similar scores to images with large differences in psychovisual quality, as illustrated in Figure 5.3. It is possible to notice that the MSE does not reflect the image quality, while the psychovisual quality degradation between the images is measured more accurately than the structural similarity index (SSIM) is considered. Specifically, all three images have almost the same MSE, but the structural similarity is more coherent to what one would expect from a comparison metric.

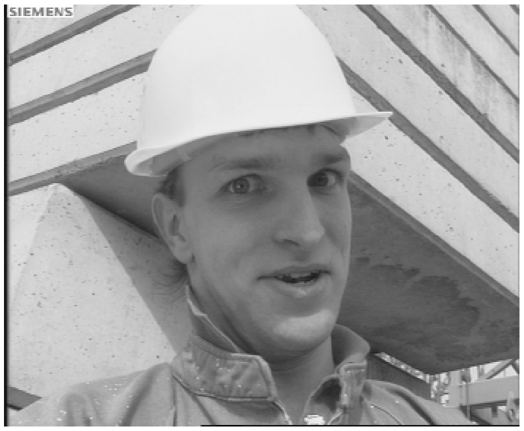
The SSIM [25] was proposed as a metric for comparing images which correlates more appropriately with the human perception. It maps two images into an index in the interval $[-1, 1]$, where higher values are given to more similar pairs of images, calculated as

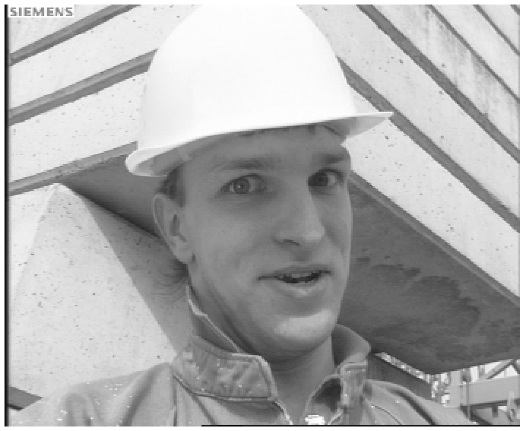
$$\mathrm{SSIM}(A,B) = \frac{(2\mu_A\mu_B + c_1)(2\sigma_{AB} + c_2)}{(\mu_A^2 + \mu_B^2 + c_1)(\sigma_A^2 + \sigma_B^2 + c_2)} \tag{5.7}$$

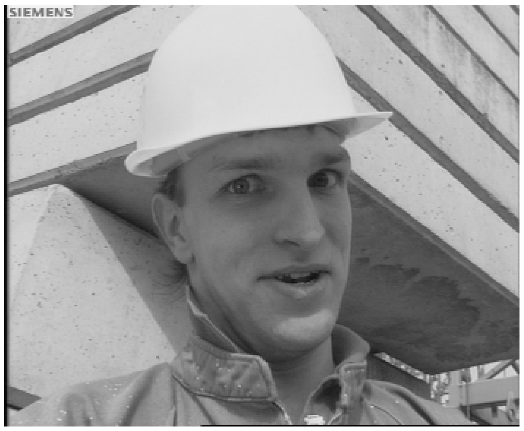
where μ_A, μ_B, σ_A^2 and σ_B^2 are the averages and variances of A and B, σ_{AB} is the covariance between A and B, and both c_1 and c_2 are predefined constants. This metric is calculated as the average of the score between several blocks using a sliding window of 11×11 pixels.

The structural dissimilarity (DSSIM) is a derived metric from the structural similarity that results in more distinct values, since a small variation in the original SSIM indicates a large difference in image quality. It is given by

$$\mathrm{DSSIM}(x,y) = \frac{1}{1 - \mathrm{SSIM}(x,y)} \tag{5.8}$$


(a) Original image (SSIM=1.0, MSE=0)


(b) Multiplied by 1.072 (SSIM=0.995837, MSE=145.96)


(c) Subtracted by 12 pixels (SSIM=0.994981, MSE=143.97)

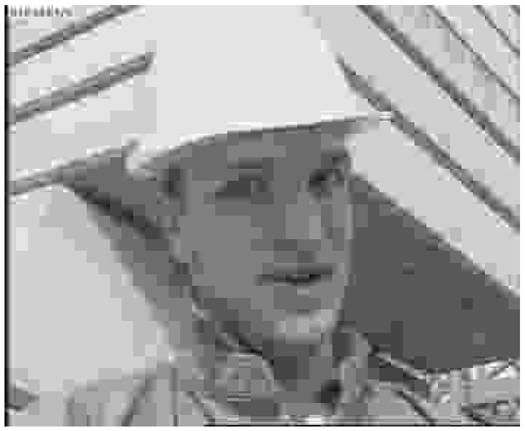

(d) Compressed by JPEG (SSIM=0.742805, MSE=142.91)

Figure 5.3: Several distorted versions of the same image with different perceptual qualities and approximately the same MSE.

5.5 Proposed Method

The video encoder described in this section can be separated into two different modules. Section 5.5.1 describes a heuristic used to decide the number, the position and the volume of the range blocks and Section 5.5.2 describes how to encode volumetric blocks of pixels using fractal codes and how to split them.

5.5.1 Rate-Distortion Heuristic

The proposed method is based on the heuristic created by Saupe et al. [22] for image compression with some adjustments to enable fast transcoding of the compressed data and replacing the mean squared error (MSE) by the sum of squared differences (SSD), therefore, the total volume of each block contributes to the distortion measure.

Initially, a group of consecutive frames is preprocessed by encoding it at a relative high bit rate by using a predefined i_{upper} value. It is subdivided into a uniform grid of range blocks with $16 \times 16 \times 16$ pixels, which are encoded and inserted into the priority queue.

The rate-distortion heuristic creates a new pair of range blocks at each iteration which replaces the one taken from the queue. Instead of destroying this parent block, the algorithm then keeps all the range blocks created during the entire process marking them with the number of the iteration in which they were created.

At every Δi iterations, the group of frames is encoded by the arithmetic encoder several times to generate a table that associates how many iterations must be considered to satisfy each desired maximum rate. This reencoding process has a very low overhead since the rate-distortion heuristic creates, at each step, several versions of the same content with different bit rates and distortions.

The transcoder reads this encoded group of frames, uses the table to choose a maximum number of iterations i_{target} that achieves the desired bit rate, and rewrites the file ignoring every block that was created after i_{target} and discarding both the iteration number associated with each block and the encoding parameters of the unused blocks.

The entire process is applied independently to each group consecutive frames allowing the target rate to be completely different for each part of the video during the transcoding. Therefore, then the target rate can fluctuate during the transmission of the video. Some encoders use Rate-Distortion Optimization [19] to adjust the rate of each group of frames in order to minimize the total distortion, but since it is impossible to preview the bandwidth of the transmission channel, this approach cannot be used in such a case.

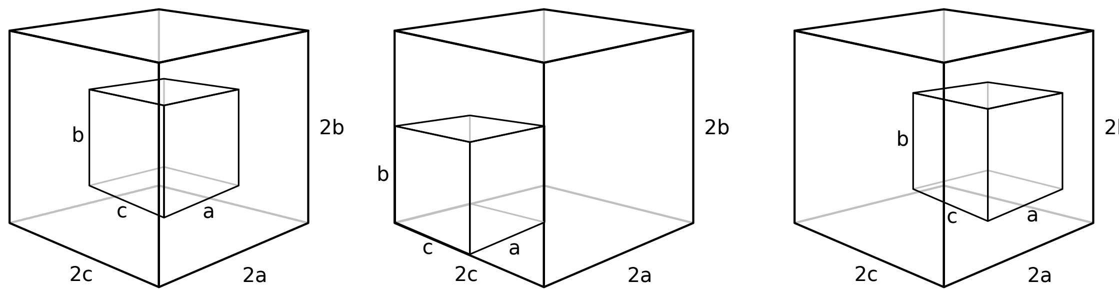


Figure 5.4: A few possible relationships between a range and a domain block in the proposed video encoder.

5.5.2 Fast Fractal Block Video Encoding

During every iteration of the rate distortion heuristic, each range block of the group of frames is encoded by a generalization to three dimensions of the fractal image encoder described in Section 5.4.3. Each range block with dimensions a, b and c at the coordinates (x, y, z) is matched against a domain block with dimensions equal to $2a$, $2b$ and $2c$ located at

$$\begin{aligned} x' &= x - a + p_x \times a/2 \\ y' &= y - b + p_y \times b/2 \\ z' &= z - c + p_z \times c/2 \end{aligned} \tag{5.9}$$

where parameters p_x, p_y and p_z must be equal to 0, 1 or 2. Range blocks with volumes smaller than 512 pixels have all these parameters set to 1. This relationship between the range and the domain block is illustrated in Figure 5.4. The domain block is transformed by Equation 5.1, before the matching, with the best α parameter chosen among the values in the set $\{0.25, 0.5, 0.75, 1.0\}$.

Each parent block is divided by choosing the direction (along horizontal, vertical or temporal axis) and the position of the cutting plane used to split it into two range blocks. This position is chosen in order to minimize the function $\sigma_A \times V_A + \sigma_B \times V_B$, where σ_A, σ_B are the variances of the resulting blocks and V_A, V_B are their volumes. The variances and averages of every block are calculated using integral volumes as described by Glassner [6].

The resulting range blocks are encoded by transmitting their α and $\bar{r}$ parameters. Range blocks with any dimension smaller than four pixels have their α parameters set to zero.

All the required symbols and parameters are encoded using a context-adaptive arithmetic coder [21]. Each range block is encoded by its α parameter, which occupies 2 bits in the worst case, along with $\bar{r}$, which is quantized according to the range block volume as shown in Table 5.1. For range blocks with one or more dimensions smaller than 2 pixels, the only transmitted parameter is $\bar{r}$.

Along with these parameters, the spatial subdivision tree for each block in the initial uniform subdivision is encoded by a sequence of symbols pointing to the decoder, in a depth-first order, whether a certain region was subdivided or not, which direction it was split and the coordinate of the splitting plane. The α parameter and the binary decision symbols in the spatial subdivision tree have their own high order adaptive contexts, one for each possible value of $\lfloor \log V \rfloor$, where V is the total volume of the encoded block. The direction in which each block is split is encoded by another set of 3 high order adaptive contexts chosen according to the direction used to split its parent.

The $\bar{r}$ parameter is encoded as the difference between a quantized prediction and the real quantized value. The prediction is calculated as the average of $\bar{r}$ of the neighboring blocks located at the top, to the left and behind the encoded block weighted by their area of contact. This difference is encoded by the Adaptive Goulomb-Rice code described in [26], using one context for each possible $\lfloor \log V \rfloor$ in the same manner as the other parameters.

Volume	1	2	4	8	16	32	64	128	256	512	1024	2048	4096
Quantization step	16	16	16	8	8	4	4	2	2	1	1	1	1
Number of used bits	4	4	4	5	5	6	6	7	7	8	8	8	8

Table 5.1: Uniform quantizers applied according to the volume of the range block.

The intermediary representation created by the encoder and supplied to the transcoding process encodes the iteration number in which each block was created using the same Adaptive Goulomb-Rice code as $\bar{r}$ and it stores the $\bar{r}$ and α for every block, including the ones that were split by the rate-distortion heuristic (which are not included in the final stream sent to the decoder).

The decoding process is accelerated by three different and complementary methods. The initial volumetric image that is used in the first iteration of the decoder is composed by filling each range block with its mean (for more details, see [17]). The used pixel intensity transform is the one proposed by Øien and Lepsøy [18] with additional proofs and details given by Pi et al. [20]. Each iteration is applied according to the Gauss-Seidel inspired method proposed by Hamzaoui [7], which uses only one image during the iterations to overwrite each range block with its updated contents. The use of these methods assures that the decoding process converges in 4 iterations or less, instead of the usual 8 to 10 iterations used by other fractal decoders.

5.6 Experimental Results

The video sequences were encoded on an Intel Core 2 Duo E6750 processor, 2.66 GHz with 3GB of RAM running the Arch Linux operating system. The method was implemented using the C++ programming language without any SIMD optimizations and compiled by GCC using the -O3 flag.

Each group had at most 32 frames in it because of precision limits in the construction of the integral volumes and to ensure that every range block in the initial partition had at least one element in its domain pool.

The proposed approach is compared to the state-of-the-art H.264 encoder, called x264 [28], using the standard grayscale benchmark sequences 'Foreman', 'Car phone', 'Bus', 'Football', 'Akiyo', 'Hall monitor', 'Bowing', and 'Miss America' in the CIF format [2]. Table 5.2 presents, for each sequence, the number of frames, the average encoding time needed to generate the intermediate file used by the transcoder[1], the resulting file size of this initial encoding, and the final file size at the maximum available bit rate in the preprocessed file (i.e. without any extra data used by the transcoding process).

The video sequences in the following experiments were encoded at low bit rates, which implies that the results had high distortions when measured by Equation 5.5. As shown in [25], the ambiguity of the metrics derived from the SSD from a perceptual point of view is high and becomes even larger as the distortion increases. It is important to observe that both α and $\bar{r}$ parameters are quantized (i.e. they must assume one of a small set of possible values instead of being continuous), which causes a mean shift and a contrast change in every range block, even though the effect of these quantizations is perceptually negligible.

[1]It is important to mention that this process is executed only once and each intermediate file is stored and used when necessary.

Sequence	# Frames	Time (s)	Size (KB)	Size without aditional data (KB)
Foreman	300	5.8	390.5	194.7
Car phone	457	6.1	392.6	200.6
Bus	150	5.4	382.6	188.0
Football	260	5.5	387.1	193.5
Akiyo	300	5.0	336.8	170.5
Hall monitor	300	5.2	366.7	187.7
Bowing	300	5.0	343.0	175.6
Miss America	179	5.8	315.9	149.1

Table 5.2: The number of frames for the video sequences, the encoding time of the proposed method and the size of each encoded file with and without the extra data used by the transcoding process.

In order to ensure a proper comparison between both methods, the mean structural dissimilarity is used in the experiments. This metric is widely accepted for its simplicity and reasonably accuracy, being employed in the design of several image encoders, such as [12], and has a built-in implementation in the x264.

The bit rate was varied to closely match the same values in both encoders. As observed in Figures 5.5 and 5.6, the proposed encoder outperforms the x264 codec at very low bit rates in high motion sequences. The motion compensation algorithm of the H.264 encoder cannot operate properly in these conditions given that an accurate prediction of each frame would require a large amount of bits.

The x264 was configured to closely match the behavior of the proposed encoder by forcing it to insert a keyframe at every 32 frames and compiling it without any CPU specific optimizations. The command line used to invoke this encoder was

```
x264 --tune ssim --preset medium --profile baseline
--keyint 32 --bitrate ’target bitrate’
```

The Rate-Distortion heuristic, described in Section 5.5.1, was configured to use 125000 iterations for each entire sequence and Δi was set to 1000 iterations. The encoding process shown in Table 5.2 is executed only once and generates the compressed video combined with extra data to allow the fast transcoding of the sequence. As shown in the last column of the table, the overhead of the extra data is quite large, almost doubling the size of the compressed video.

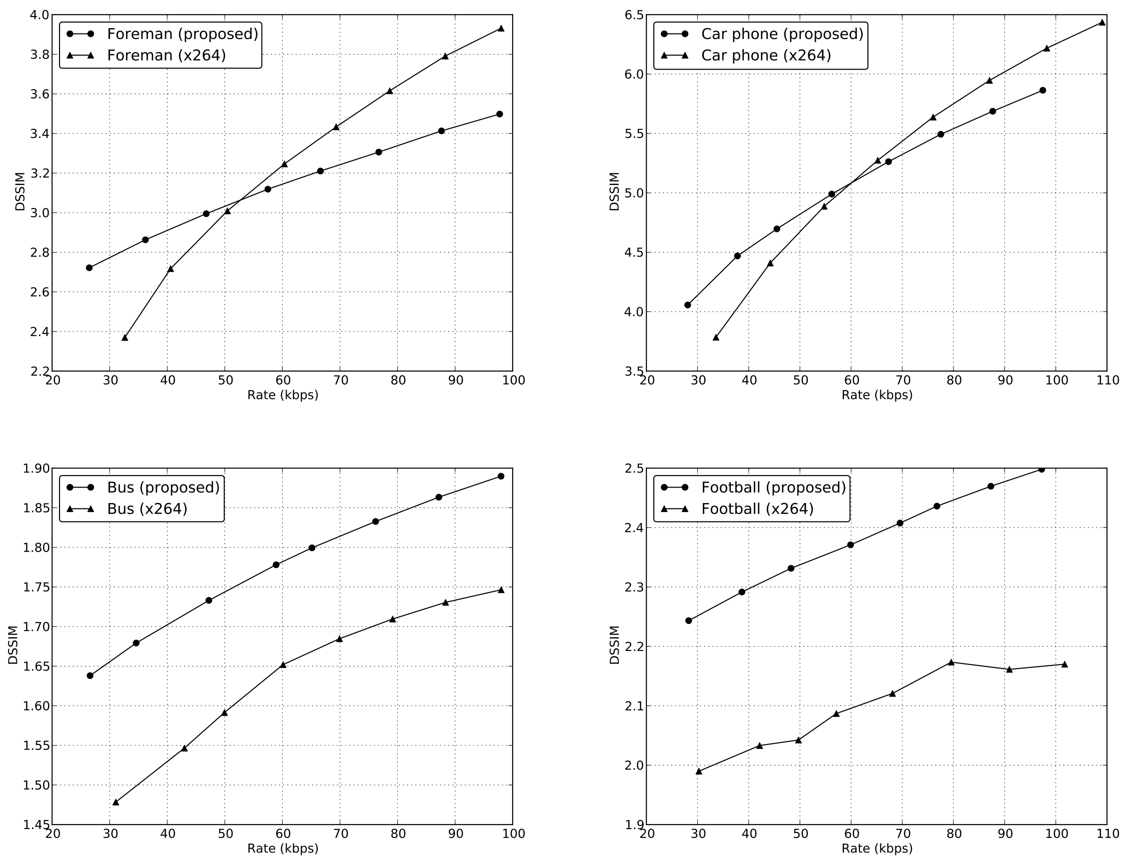


Figure 5.5: Mean structural dissimilarity at different rates for the proposed video encoder and the x264 encoder.

Figures 5.7 and 5.8 show the speed of the transcoding process. The required time to transcode the sequences is at most 140 ms at the highest bit rates. Regardless the preprocessing step, which is performed only once and can be considered as an offline process, the proposed method is much faster than directly encoding the video with x264.

The transcoding algorithm is limited only by how fast the data can be read and written since the total transcoding time is linearly correlated with the target rate. Figures 5.9 and 5.10 show that the rate control is precise, achieving the desired constraint with a negligible error.

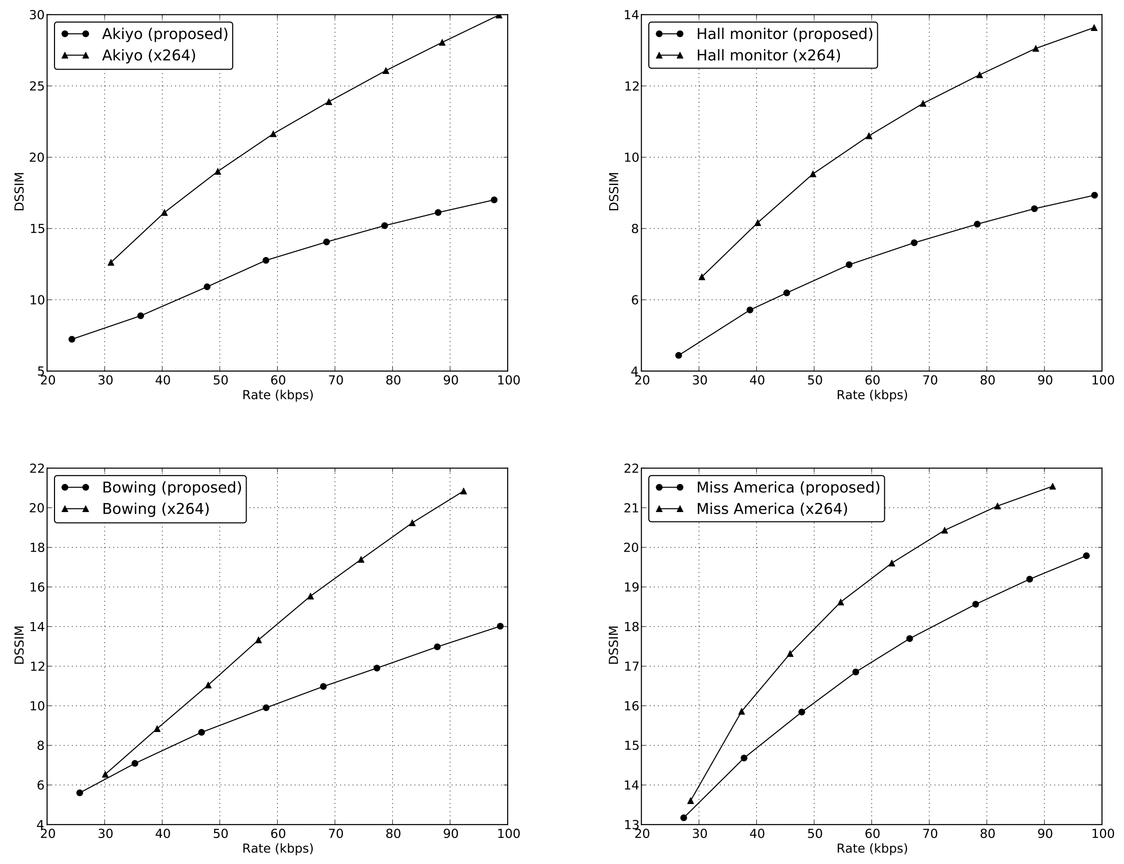


Figure 5.6: Mean structural dissimilarity at different rates for the proposed video encoder and the x264 encoder.

5.7 Conclusions and Future Work

The proposed approach can rapidly transcode a preprocessed video sequence into a large range of different bit rates with extreme fine control of the resulting rate. It employs a fast fractal encoding method using volumetric range and domain blocks matched against each other using a generalization of a fast fractal encoder to three dimensions.

It is important to mention that the proposed rate control and the transcoding heuristic could be applied to other encoding methods that are not based on fractals but are still adaptive. The near-optimal behavior of the transcoding algorithm,

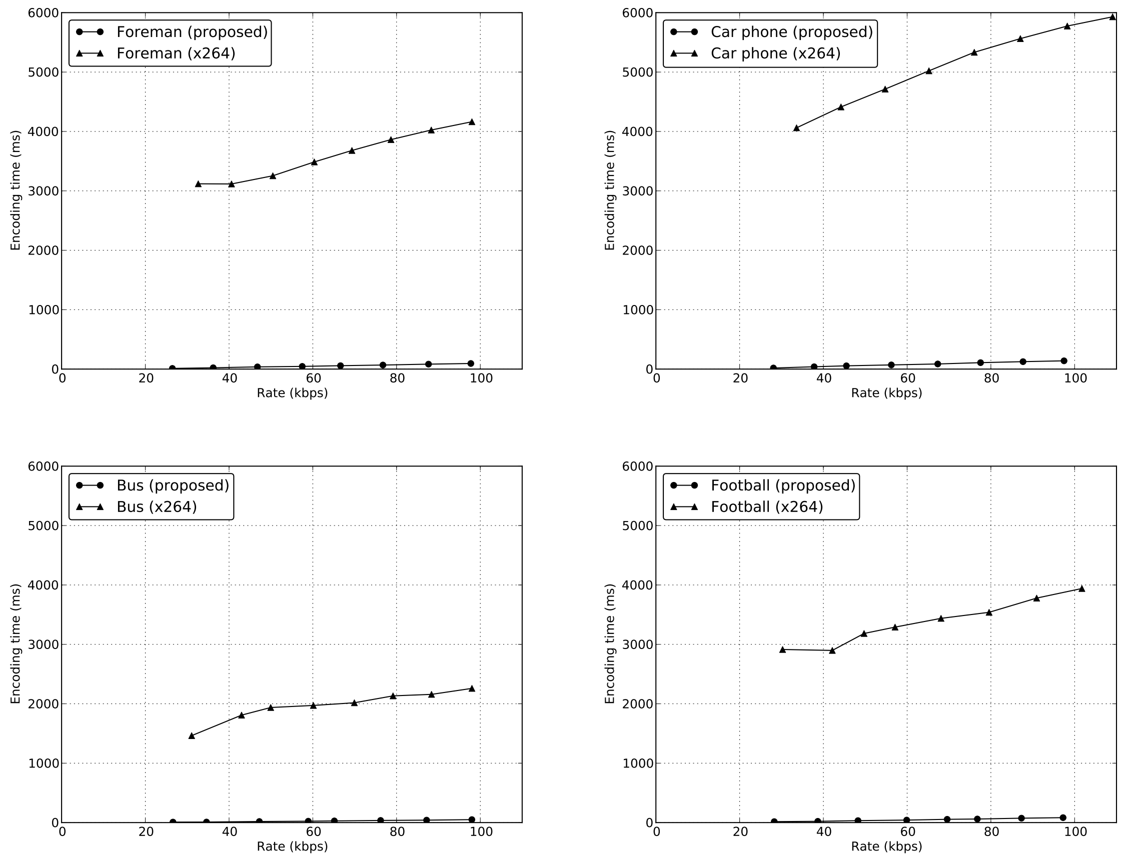


Figure 5.7: Encoding time at different rates for the video transcoder and the x264 encoder.

combined with better block encoding methods, could result in viable alternative to the current commonly used video encoders.

5.8 References

[1] A. Chabarchine and R. Creutzburg. 3D Fractal Compression for Real-Time Video. In *2nd International Symposium on Image and Signal Processing and Analysis*, pages 570–573, 2001.

[2] CIPR. Sequences. `http://www.cipr.rpi.edu/resource/sequences/`.

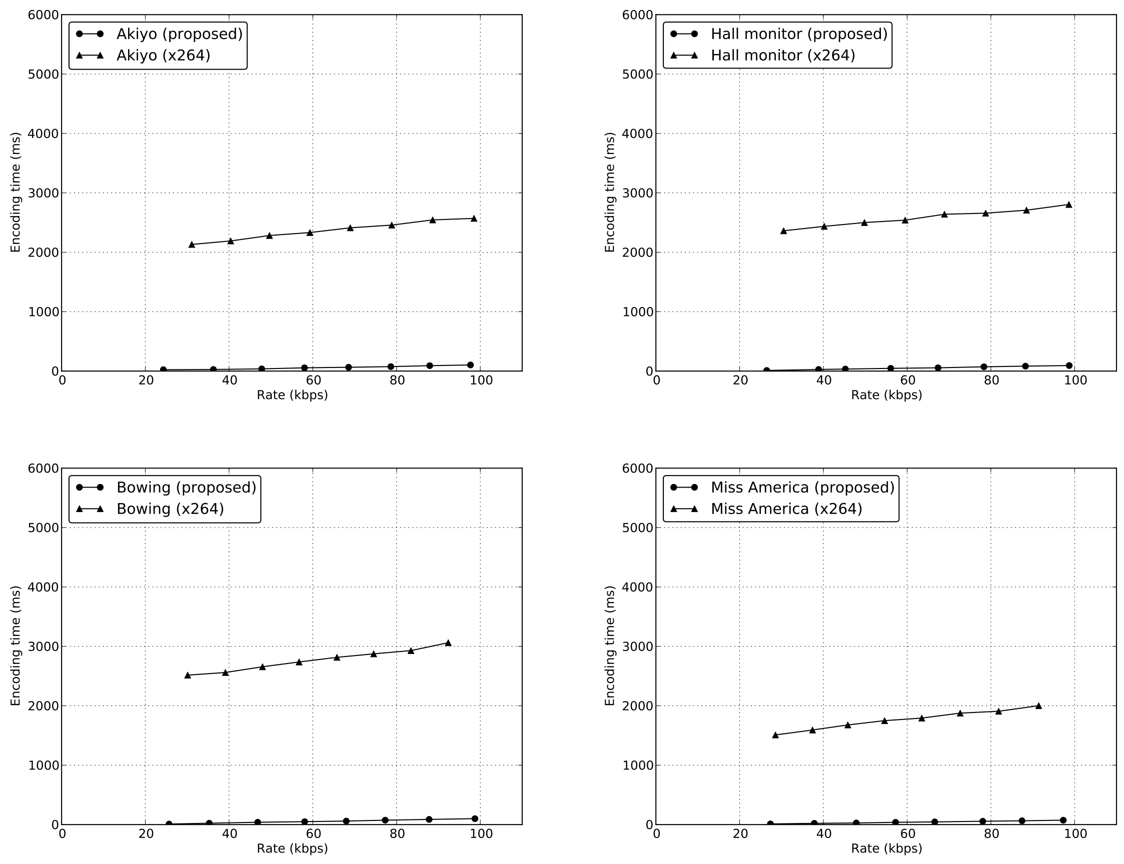
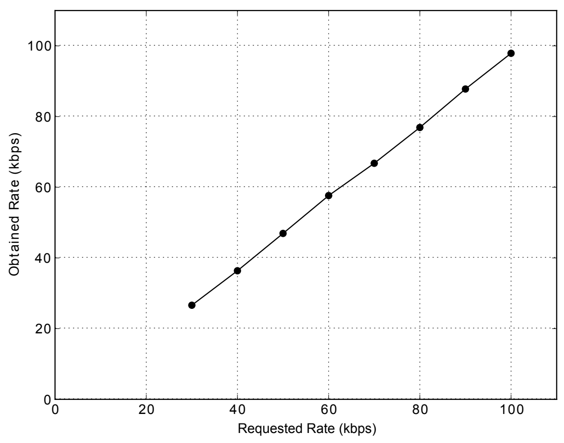


Figure 5.8: Encoding time at different rates for the video transcoder and the x264 encoder.

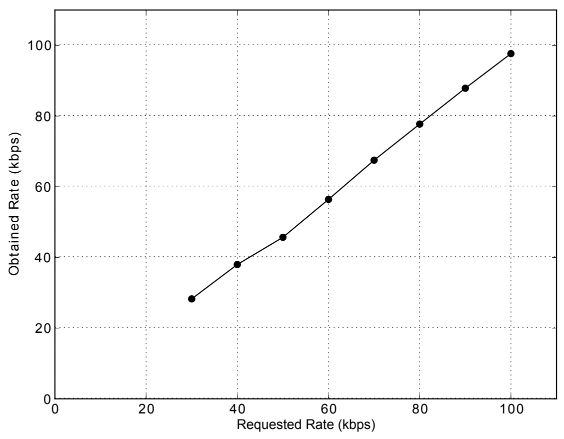
[3] V. de Lima, W. R. Schwartz, and H. Pedrini. Fast Low Bit-Rate 3D Searchless Fractal Video Encoding. In *Conference on Graphics, Patterns and Images*, Maceio, AL, Brazil, 2011.

[4] Y. Fisher, D.N. Rogovin, and T.J. Shen. Fractal (Self-VQ) Encoding of Video Sequences. *Visual Communications and Image Processing*, 2308(1):1359–1370, 1994.

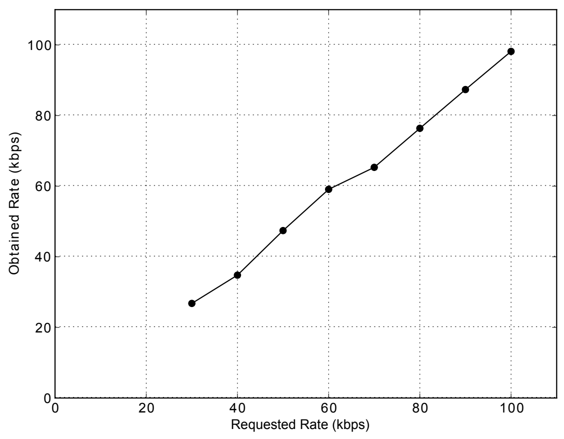
[5] S. Furao and O. Hasegawa. A Fast No Search Fractal Image Coding Method. *Signal Processing: Image Communication*, 19(5):393–404, 2004.



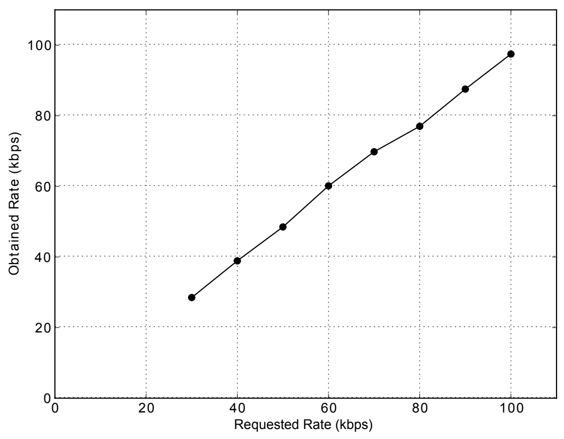
(a) Foreman



(b) Car phone



(c) Bus



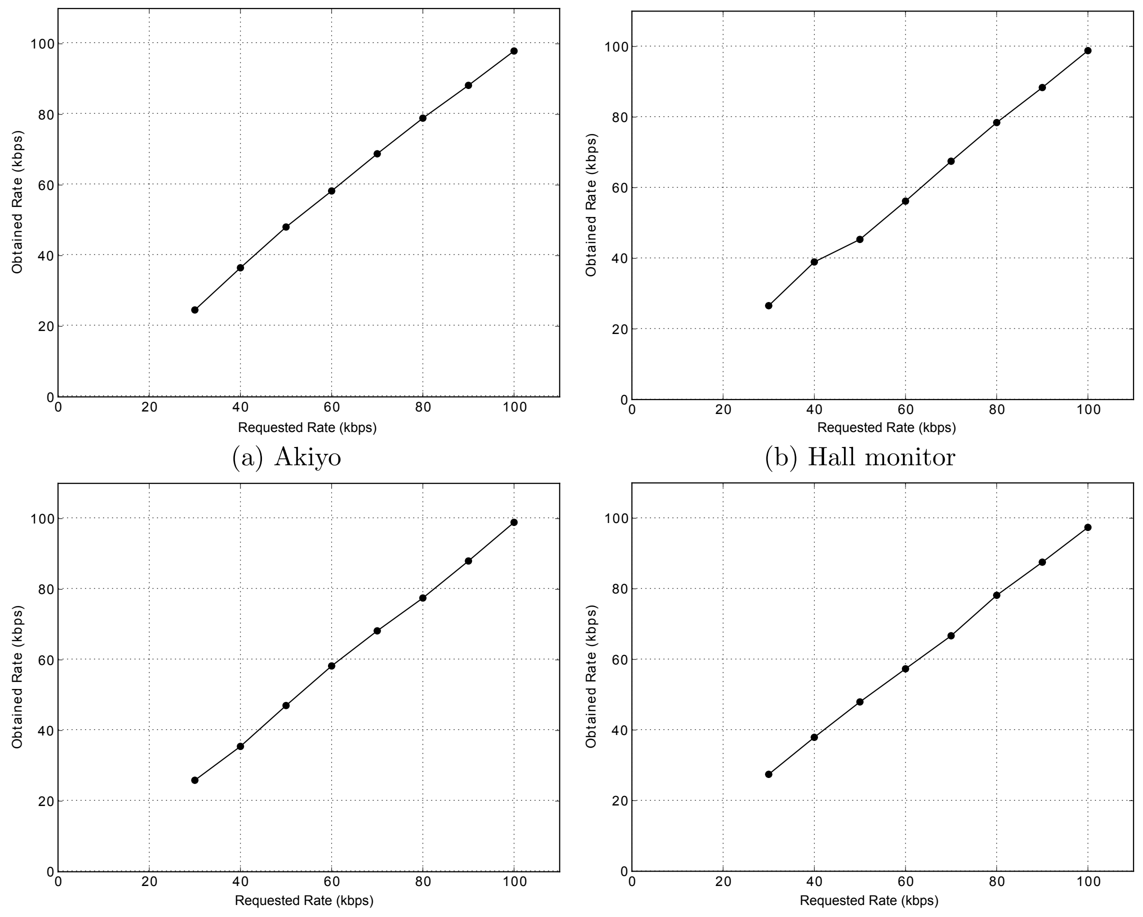
(d) Football

Figure 5.9: Requested and obtained rates for the proposed method.

[6] Andrew S. Glassner. Graphics Gems. In Andrew S. Glassner, editor, *Multidimensional Sum Tables*, pages 376–381. Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[7] Raouf Hamzaoui. Fast Iterative Methods for Fractal Image Compression. *Journal of Mathematical Imaging and Vision*, 11:147–159, October 1999.

[8] L.P. Hurd, M.A. Gustavus, and M.F. Barnsley. Fractal Video Compression. In *Thirty-Seventh IEEE Computer Society International Conference*, pages 41–42, February 1992.



(a) Akiyo (b) Hall monitor

(c) Bowing (d) Miss America

Figure 5.10: Requested and obtained rates for the proposed method.

[9] I. Ishfaq Ahmad, Xiaohui Wei, Yu Sun, and Ya-Qin Zhang. Video Transcoding: An Overview of Various Techniques and Research Issues. *IEEE Transactions on Multimedia*, 7:793–804, 2005.

[10] A.E. Jacquin. Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations. *IEEE Transactions on Image Processing*, 1(1):18–30, January 1992.

[11] C. Kim, R. Kim, and S. Lee. Fractal Coding of Video Sequence using Circular Prediction Mapping and Noncontractive Interframe Mapping. *IEEE Transactions on Image Processing*, 7(4):601–605, April 1998.

[12] P. K. Krause. FTC - Floating Precision Texture Compression. *Computers and Graphics*, 34:594–601, October 2010.

[13] M.S. Lazar and L.T. Bruton. Fractal Block Coding of Digital Video. *IEEE Transactions on Circuits and Systems for Video Technology*, 4(3):297–308, June 1994.

[14] H. Li, M. Novak, and R. Forchheimer. Fractal-Based Image Sequence Compression Scheme. *Optical Engineering*, 32(7):1588–1595, 1993.

[15] V. Lima, W. R. Schwartz, and H. Pedrini. Fast Adaptive Transcoding Based on a 3D Low Bit Rate Fractal Video Encoder. *IEEE Transactions on Multimedia (submetido)*, 2011.

[16] V. Lima, W. R. Schwartz, and H. Pedrini. Fractal Image Encoding Using a Constant Size Domain Pool. In *Workshop de Visão Computacional*, pages 137–142, 2011.

[17] Yong Ho Moon, Hyung Soon Kim, and Jae Ho Kim. A Fast Fractal Decoding Algorithm based on the Selection of an Initial Image. *IEEE Transactions on Image Processing*, 9(5):941–945, May 2000.

[18] G.E. Øien and S. Lepsøy. *A Class of Fractal Image Coders with Fast Decoder Convergence*, chapter Fractal Image Compression, pages 153–175. Springer-Verlag, London, UK, 1995.

[19] A. Ortega and K. Ramchandram. Rate-Distortion Methods for Image and Video Compression. *IEEE Signal Processing Magazine*, 15(6):23–50, 1998.

[20] M. Pi, A. Basu, and M. Mandal. A New Decoding Algorithm based on Range Block Mean and Contrast Scaling. In *International Conference on Image Processing*, volume 2, pages II – 271–4 vol.3, September 2003.

[21] A. Said. *Lossless Compression Handbook*, chapter Arithmetic Coding. Communications, Networking, and Multimedia. Academic Press, 2003.

[22] D. Saupe, M. Ruhl, R. Hamzaoui, L. Grandi, and D. Marini. Optimal Hierarchical Partitions for Fractal Image Compression. In *IEEE International Conference on Image Processing*, pages 737–741, Chicago, IL, USA, October 1998.

[23] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103–1120, September 2007.

[24] C.S. Tong and M. Pi. Fast Fractal Image Encoding based on Adaptive Search. *IEEE Transactions on Image Processing*, 10(9):1269–1277, September 2001.

[25] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600 –612, April 2004.

[26] M.J. Weinberger, G. Seroussi, and G. Sapiro. LOCO-I: A Low Complexity, Context-based, Lossless Image Compression Algorithm. *Data Compression Conference*, pages 140–149, 1996.

[27] X. Wu, D.J. Jackson, and H. Chen. Novel Fractal Image-Encoding Algorithm Based on a Full-Binary-Tree Searchless Iterated Function System. *Optical Engineering*, 44(10), 2005.

[28] x264. Video Encoder. `http://www.videolan.org/developers/x264.html`.

[29] Z. Yao and R. Wilson. Hybrid 3D Fractal Coding with Neighbourhood Vector Quantisation. *EURASIP Journal on Applied Signal Processing*, 2004:2571–2579, January 2004.

[30] Shiping Zhu, Zaikuo Wang, and K. Belloulata. A Novel Fractal Monocular and Stereo Video Codec based on MCP and DCP. In *IEEE International Conference on Industrial Technology*, pages 168–172, March 2010.

Capítulo 6

Conclusões

O codificador baseado em *Matching Pursuits* do artigo *A Very Low Bit-Rate Minimalist Video Encoder Based on Matching Pursuits* [11] se mostrou competitivo com um padrão destinado a aplicações de videoconferência, o H.263, entretanto, a qualidade de imagem obtida não justificou o custo computacional extremamente elevado. Outros problemas sérios desta abordagem são dificuldades na quantização dos coeficientes, no controle da taxa de bits resultante e na construção do dicionário.

O método fractal rápido proposto em *Fractal Image Encoding Using a Constant Size Domain Pool* [31] é levemente superior em qualidade aos métodos sem busca e causa menos artefatos visuais, entretanto, apresenta um custo computacional maior. Apesar disso, o tempo de codificação permanece na mesma ordem de grandeza de outros métodos eficientes. O método de construção de colagens com busca em um *domain pool* de tamanho constante foi reutilizado no quarto artigo.

O codificador de vídeo apresentado em *Fast Low Bit-Rate 3D Searchless Fractal Video Encoding* [10] é competitivo com o estado da arte tanto no tempo de codificação quanto, em casos em que há muito movimento e uma alta taxa de compressão, na qualidade perceptual da imagem.

No quarto artigo [30] é proposto um codificador capaz de alterar a taxa de transmissão do vídeo em tempo real após o pré-processamento dos dados. Ele inclui um conjunto maior de candidatos nos casamentos entre blocos moldes e domínios. O algoritmo de subdivisão do espaço particiona os blocos de forma a minimizar o produto entre a variância e o volume das subdivisões resultantes. As médias dos blocos moldes são estimadas por meio de um algoritmo mais complexo.

Trabalhos futuros incluem o uso de *Rate Distortion Optimization* [39] para controlar tanto o tamanho resultante do arquivo codificado quanto a poda dos blocos da árvore de subdivisão do espaço e a quantização de todos os coeficientes. Métodos mais elaborados de quantização também poderiam ser usados nos coeficientes de cada molde.

Referências Bibliográficas

[1] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, November 2006.

[2] K. Aizawa, H. Harashima, and T. Saito. Model-Based Analysis Synthesis Image Coding (MBASIC) System for a Person's Face. *Signal Processing: Image Communications*, 1(2):139–152, October 1989.

[3] O.K Al-Shaykh, E. Miloslavsky, T. Nomura, R. Neff, and A. Zakhor. Video Compression using Matching Pursuits. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(1):123–143, February 1999.

[4] avcodec. libavcodec: A Library containing Decoders and Encoders for Audio/Video Codecs, 2010. `http://www.ffmpeg.org/`.

[5] V. Bhaskaran and K. Konstantinides. *Image and Video Compression Standards: Algorithms and Architectures.* Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[6] CCITT. Video Codec for Audiovisual Services at p $\times$ 64 kbit/s, CCITT Recommendation H.261, CDM XV-R 37-E, August 1990.

[7] A. Chabarchine and R. Creutzburg. 3D Fractal Compression for Real-Time Video. In *2nd International Symposium on Image and Signal Processing and Analysis*, pages 570–573, 2001.

[8] CIPR. Sequences. `http://www.cipr.rpi.edu/resource/sequences/`.

[9] G. Davis. *Adaptive Nonlinear Approximations.* PhD thesis, Department of Mathematics, New York University, 1994.

[10] V. de Lima, W. R. Schwartz, and H. Pedrini. Fast Low Bit-Rate 3D Searchless Fractal Video Encoding. In *Conference on Graphics, Patterns and Images*, Maceio, AL, Brazil, 2011.

[11] Vitor De Lima and Helio Pedrini. A Very Low Bit-rate Minimalist Video Encoder Based on Matching Pursuits. In *Proceedings of the 15th Iberoamerican congress conference on Progress in pattern recognition, image analysis, computer vision, and applications*, CIARP'10, pages 176–183, Berlin, Heidelberg, 2010. Springer-Verlag.

[12] M. Elad and M. Aharon. Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745, December 2006.

[13] Y. Fisher and S. Menlove. Fractal Encoding with HV Partitions. In Y. Fisher, editor, *Fractal Image Compression - Theory and Application*, pages 119–126. Springer-Verlag, London, UK, 1995.

[14] Y. Fisher, D.N. Rogovin, and T.J. Shen. Fractal (Self-VQ) Encoding of Video Sequences. *Visual Communications and Image Processing*, 2308(1):1359–1370, 1994.

[15] Yuval Fisher. *Fractal Image Compression: Theory and Application.* Springer-Verlag, London, UK, 1995.

[16] S. Furao and O. Hasegawa. A Fast No Search Fractal Image Coding Method. *Signal Processing: Image Communication*, 19(5):393–404, 2004.

[17] B. Furht and B. Furht. *Motion Estimation Algorithms for Video Compression.* Kluwer Academic Publishers, Norwell, MA, USA, 1996.

[18] Andrew S. Glassner. Graphics Gems. In Andrew S. Glassner, editor, *Multidimensional Sum Tables*, pages 376–381. Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[19] S. Golomb. Run-length encodings (corresp.). *Information Theory, IEEE Transactions on*, 12(3):399–401, July 1966.

[20] H263. ITU-T Recommendation H.263, Video Coding for Low Bit Rate Communication, September 1997.

[21] Raouf Hamzaoui. Fast Iterative Methods for Fractal Image Compression. *Journal of Mathematical Imaging and Vision*, 11:147–159, October 1999.

[22] D.A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098 –1101, September 1952.

[23] L.P. Hurd, M.A. Gustavus, and M.F. Barnsley. Fractal Video Compression. In *Thirty-Seventh IEEE Computer Society International Conference*, pages 41–42, February 1992.

[24] I. Ishfaq Ahmad, Xiaohui Wei, Yu Sun, and Ya-Qin Zhang. Video Transcoding: An Overview of Various Techniques and Research Issues. *IEEE Transactions on Multimedia*, 7:793–804, 2005.

[25] A.E. Jacquin. Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations. *IEEE Transactions on Image Processing*, 1(1):18–30, January 1992.

[26] C. Kim, R. Kim, and S. Lee. Fractal Coding of Video Sequence using Circular Prediction Mapping and Noncontractive Interframe Mapping. *IEEE Transactions on Image Processing*, 7(4):601–605, April 1998.

[27] P. K. Krause. FTC - Floating Precision Texture Compression. *Computers and Graphics*, 34:594–601, October 2010.

[28] M.S. Lazar and L.T. Bruton. Fractal Block Coding of Digital Video. *IEEE Transactions on Circuits and Systems for Video Technology*, 4(3):297–308, June 1994.

[29] H. Li, M. Novak, and R. Forchheimer. Fractal-Based Image Sequence Compression Scheme. *Optical Engineering*, 32(7):1588–1595, 1993.

[30] V. Lima, W. R. Schwartz, and H. Pedrini. Fast Adaptive Transcoding Based on a 3D Low Bit Rate Fractal Video Encoder. *IEEE Transactions on Multimedia (submetido)*, 2011.

[31] V. Lima, W. R. Schwartz, and H. Pedrini. Fractal Image Encoding Using a Constant Size Domain Pool. In *Workshop de Visão Computacional*, pages 137–142, 2011.

[32] S. Mallat and Z. Zhang. Matching Pursuit with Time-Frequency Dictionaries. *IEEE Transactions on Signal Processing*, 41:3397–3415, December 1993.

[33] Xiph.org Test Media. Video Sequences, 2010. `http://media.xiph.org/video/derf/`.

[34] Yong Ho Moon, Hyung Soon Kim, and Jae Ho Kim. A Fast Fractal Decoding Algorithm based on the Selection of an Initial Image. *IEEE Transactions on Image Processing*, 9(5):941–945, May 2000.

[35] R. Neff, T. Nomura, and A. Zakhor. Decoder Complexity and Performance Comparison of Matching Pursuit and DCT-based MPEG-4 Video Codecs. In *International Conference on Image Processing*, pages 783–787, Chicago, IL, USA, October 1998.

[36] R. Neff and A. Zakhor. Very-Low Bit-Rate Video Coding Based on Matching Pursuits. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(1):158–171, February 1997.

[37] NVIDIA. CUDA - Parallel Computing Architecture, 2010. `http://www.nvidia.com/`.

[38] G.E. Øien and S. Lepsøy. *A Class of Fractal Image Coders with Fast Decoder Convergence*, chapter Fractal Image Compression, pages 153–175. Springer-Verlag, London, UK, 1995.

[39] A. Ortega and K. Ramchandram. Rate-Distortion Methods for Image and Video Compression. *IEEE Signal Processing Magazine*, 15(6):23–50, 1998.

[40] M. Pi, A. Basu, and M. Mandal. A New Decoding Algorithm based on Range Block Mean and Contrast Scaling. In *International Conference on Image Processing*, volume 2, pages II – 271–4 vol.3, September 2003.

[41] L. Rebollo-Neira and D. Lowe. Optimized Orthogonal Matching Pursuit Approach. *IEEE Signal Processing Letters*, 9(4):137–140, April 2002.

[42] R. Rice and J. Plaunt. Adaptive Variable-Length Coding for Efficient Compression of Spacecraft Television Data. *Communication Technology, IEEE Transactions on*, 19(6):889 –897, December 1971.

[43] J. Rissanen and G. G. Langdon. Arithmetic coding. *IBM Journal of Research and Development*, 23(2):149 –162, March 1979.

[44] M. Ruhl and H. Hartenstein. Optimal Fractal Coding is NP-Hard. In *Data Compression Conference*, pages 261–270, March 1997.

[45] A. Said. *Lossless Compression Handbook*, chapter Arithmetic Coding. Communications, Networking, and Multimedia. Academic Press, 2003.

[46] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison Wesley, Reading MA, 1990.

[47] D. Saupe, M. Ruhl, R. Hamzaoui, L. Grandi, and D. Marini. Optimal Hierarchical Partitions for Fractal Image Compression. In *IEEE International Conference on Image Processing*, pages 737–741, Chicago, IL, USA, October 1998.

[48] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103–1120, September 2007.

[49] D. Sculley and C.E. Brodley. Compression and Machine Learning: A New Perspective on Feature Space Vectors. In *Data Compression Conference*, pages 332–332, Snowbird, UT, USA, March 2006.

[50] C. E. Shannon. Prediction and entropy of printed English. *Bell Systems Technical Journal*, 30:50–64, 1951.

[51] J. Shapiro. Application of the Embedded Wavelet Hierarchical Image Coder to Very Low Bit Rate Image Coding. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 558–561, Minneapolis, MN, USA, April 1993.

[52] I. Sutskover and D. Malah. Hierarchical Fast Decoding of Fractal Image Representation using Quadtree Partitioning. In *International Conference on Image Processing and Its Applications*, volume 2, pages 581–585, 1999.

[53] J. Teuhola. Fast Image Compression by Quadtree Prediction. *Real-Time Imaging*, 4:299–308, August 1998.

[54] C.S. Tong and M. Pi. Fast Fractal Image Encoding based on Adaptive Search. *IEEE Transactions on Image Processing*, 10(9):1269–1277, September 2001.

[55] B. Wang, Y. Wang, and P. Yin. A Two Pass H.264-Based Matching Pursuit Video Coder. In *IEEE International Conference on Image Processing*, pages 3149–3152, Atlanta, GA, USA, October 2006.

[56] X. Wang, Y. Wang, and J. Yun. An Improved No-search Fractal Image Coding Method Based on a Fitting Plane. *Image and Vision Computing*, 28:1303–1308, August 2010.

[57] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600 –612, April 2004.

[58] M.J. Weinberger, G. Seroussi, and G. Sapiro. LOCO-I: A Low Complexity, Context-based, Lossless Image Compression Algorithm. *Data Compression Conference*, pages 140–149, 1996.

[59] X. Wu, D.J. Jackson, and H. Chen. Novel Fractal Image-Encoding Algorithm Based on a Full-Binary-Tree Searchless Iterated Function System. *Optical Engineering*, 44(10), 2005.

[60] x264. Video Encoder. `http://www.videolan.org/developers/x264.html`.

[61] Z. Yao and R. Wilson. Hybrid 3D Fractal Coding with Neighbourhood Vector Quantisation. *EURASIP Journal on Applied Signal Processing*, 2004:2571–2579, January 2004.

[62] H. Zhang, X. Wang, W. Huo, and D.M. Monro. A Hybrid Video Coder Based on H.264 with Matching Pursuits. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 889–892, Toulouse, France, July 2006.

[63] Shiping Zhu, Zaikuo Wang, and K. Belloulata. A Novel Fractal Monocular and Stereo Video Codec based on MCP and DCP. In *IEEE International Conference on Industrial Technology*, pages 168–172, March 2010.

Publicações do Autor

- V. Lima, H. Pedrini. *A Very Low Bit-Rate Minimalist Video Encoder Based on Matching Pursuits.* Lecture Notes in Computer Science, ISBN 0302-9743, vol. 6419, pp. 176-183, Springer-Verlag. Artigo apresentado no *15th Iberoamerican Congress on Pattern Recognition (CIARP'2010)*, São Paulo-SP, Brasil, 08 a 11 de Novembro de 2010.

- V. Lima, W.R. Schwartz, H. Pedrini. *Fractal Image Encoding Using a Constant Size Domain Pool.* VII Workshop de Visão Computacional (WVC'2011), Curitiba-PR, Brasil, pp. 137-142, 22 a 25 de Maio de 2011.

- V. Lima, W.R. Schwartz, H. Pedrini. *Fast Low Bit-Rate 3D Searchless Fractal Video Encoding.* Conference on Graphics, Patterns and Images (XXIV SIBGRAPI), Maceió-AL, Brasil, 28 a 31 de Agosto de 2011.

- V. Lima, W.R. Schwartz, H. Pedrini. *Fast Adaptive Transcoding Based on a 3D Low Bit Rate Fractal Video Encoder.* Artigo submetido ao *IEEE Transactions on Multimedia.*

- V. Lima, W.R. Schwartz, H. Pedrini. *3D Searchless Fractal Video Encoding at Low Bit Rates.* Artigo convidado para extensão do trabalho publicado no XXIV SIBGRAPI e submissão ao *Journal of Mathematical Imaging and Vision.*

- V. Lima, W.R. Schwartz, H. Pedrini. *Codificação Fractal de Imagens Baseada em Busca Local.* Artigo convidado para extensão do trabalho publicado no WVC'2011 e submissão como capítulo de livro Avanços em Visão Computacional, Editora Omnipax.