The South African Institute for Computer Scientists and
Information Technologists

ANNUAL RESEARCH AND DEVELOPMENT
SYMPOSIUM

23-24 NOVEMBER 1998
CAPE TOWN
Van Riebeek hotel in Gourdon Bay

Hosted by the University of Cape Town in association with the CSSA,
Forchesitsetam University for CHE and
The University of Natal

PROCEEDINGS

EDITED BY
D. PETKOV AND L. VENTER

SPONSORED BY

ABSA Group
The South African Institute for Computer Scientists and Information Technologists

ANNUAL RESEARCH AND DEVELOPMENT SYMPOSIUM

23-24 NOVEMBER 1998
CAPE TOWN
Van Riebeeck hotel in Gordons Bay

Hosted by the University of Cape Town in association with the CSSA, Potchefstroom University for CHE and The University of Natal

GENERAL CHAIR : PROF G. HATTINGH, PU CHE

PROGRAMME CO-CHAIRS:
PROF. L VENTER, PU CHE (Vaal Triangle), PROF. D. PETKOV, UN-PMB

LOCAL ORGANISING CHAIR: PROF. P. LICKER, UCT - IS

PROCEEDINGS

EDITED BY
D. PETKOV AND L. VENTER

SYMPOSIUM THEME:
Development of a quality academic CS/IS infrastraucture in South Africa

SPONSORED BY
FOREWORD

The South African Institute for Computer Scientists and Information Technologists (SAICSIT) promotes the cooperation of academics and industry in the area of research and development in Computer Science, Information Systems and Technology and Software Engineering. The culmination of its activities throughout the year is the annual research symposium. This book is a collection of papers presented at the 1998 such event taking place on the 23rd and 24th of November in Gordons Bay, Cape Town. The Conference is hosted by the Department of Information Systems, University of Cape Town in cooperation with the Department of Computer Science, Potchefstroom University for CHE and and Department of Computer Science and Information Systems of the University of Natal, Pietermaritzburg.

There are a total of 46 papers. The speakers represent practitioners and academics from all the major Universities and Technikons in the country. The number of industry based authors has increased compared to previous years.

We would like to express our gratitude to the referees and the paper contributors for their hard work on the papers included in this volume. The Organising and Programme Committees would like to thank the keynote speaker, Prof M.C Jackson, Dean, University of Lincolnshire and Humberside, United Kingdom, President of the International Federation for Systems Research as well as the Computer Society of South Africa and The University of Cape Town for the cooperation as well as the management and staff of the Potchefstroom University for CHE and the University of Natal for their support and for making this event a success.

Giel Hattingh, Paul Licker, Lucas Venter and Don Petkov
Table of Contents

Lynette Drevin: Activities of IFIP wg 11.8 (computer security education) & IT related ethics education in Southern Africa 1

Reinhardt A. Botha and Jan H.P. ElofT: exA Security Interpretation of the Workflow Reference Model 3

Willem Krige and Rossouw von Solms: Effective information security monitoring using data logs 9


Carl Papenfus and Reinhardt A. Botha: A shell-based approach to information security 15

Walter Smuts: A 6-Dimensional Security Classification for Information 20

Philip Machanick and Pierre Salverda: Implications of emerging DRAM technologies for the RAM page Memory hierarchy 27

Susan Brown: Practical Experience in Running a Virtual Class to Facilitate On-Campus Under Graduate Teaching 41

H.D. Masethe, T.A Dandadzi: Quality Academic Development of CS/IS Infrastructure in South Africa 49

Philip Machanick: The Skills Hierarchy and Curriculum 54

Theda Thomas: Handling diversity in Information Systems and Computer Science Students: A social Constructivist Perspective 63

Udo Averweg and G J Erwin: Critical success factors for implementation of Decision support systems 70

Magda Huisman: A conceptual model for the adoption and use of case technology 78

Paul S. Licker: A Framework for Information Systems and National Development Research 79

K. Niki Kunene and Don Petkov: On problem structuring in an Electronic Brainstorming (EBS) environment 89
Derek Smith: Characteristics of high-performing Information Systems Project Managers and Project Teams  

Lucas Venet: INSTOP: Experiences in building a multimedia application  

Scott Hazelhurst, Anton Fatti, and Andrew Henwood: Binary Decision Diagram Representations of Firewall and Router Access Lists  

Andre Joubert and Annelie Jordaan: Hardware System interfacing with Delphi 3 to achieve quality academic integration between the fields of Computer Systems and Software Engineering  

Borislav Roussev: Experience with Java in an Advanced Operating Systems Module  

Conrad Mueller: A Static Programming Paradigm  

Sipho Langa: Management Aspects of Client/Server Computing  

T Nepal and T Andrew: An Integrated Research Programme in AI applied to Telecommunications at ML Sultan Technikon  

Yuri Velinov: Electronic lectures for the mathematical subjects in Computer Science  

Philip Machanick: Disk delay lines  

D Petkov and O Petkova: One way to make better decisions related to IT Outsourcing  

Jay van Zyl: Quality Learning, Learning Quality  

Matthew O Adigun: A Case for Reuse Technology as a CS/IS Training Infrastructure  

Andy Bytheway and Grant Hearn: Academic CS/IS Infrastructure in South Africa: An exploratory stakeholder perspective  

Chantel van Niekerk: The Academic Institution and Software Vendor Partnership  

Christopher Chalmers: Quality aspects of the development of a rule-based architecture  

Rudi Harmse: Managing large programming classes using computer mediated communication and cognitive modelling techniques
Michael Muller: How to gain Quality when developing a Repository Driven User Interface

Elsabe Cloete and Lucas Venter: Reducing Fractal Encoding Complexities

Jean Bilbrough and Ian Sanders: Partial Edge Visibility in Linear Time

Philip Machanick: Design of a scalable Video on Demand architecture

Freddie Janssen: Quality considerations of Real Time access to Multidimensional Matrices

Machiel Kruger and Giel Hattingh: A Partitioning Scheme for Solving the Exact \( k \)-item 0-1 Knapsack Problem

Ian Sanders: Non-orthogonal Ray Guarding

Fanie Terblanche and Giel Hattingh: Response surface analysis as a technique for the visualization of linear models and data

Olga Petkova and Dewald Roode: A pluralist systemic framework for the evaluation of factors affecting software development productivity

Peter Warren and Marcel Viljoen: Design patterns for user interfaces

Andre de Waal and Giel Hattingh: Refuting conjectures in first order theories

Edna Randiki: Error analysis in Selected Medical Devices and Information Systems
Reducing Fractal Encoding Complexities

E Cloete * L M Venter †

*Dept. Computer Science and Information Systems, UNISA, Pretoria, 0001 Dept. Computer Science and Information Systems, PU for CHE, Vanderbijlpark, 1900

Abstract

In this paper we address the time complexity problem associated with fractal image coding. In particular, we describe a new hybrid technique called Fractal Vector Quantization coding (FVQ), which takes advantage of the best qualities in fractal coding and vector quantization (VQ).

In our proposed approach, VQ is used to construct a set of real world building blocks which can be used to approximate an arbitrary image. Fractal coding is then employed to fractalize the building blocks by finding an affine transformation for each block which best represents that block. The real world building blocks with their fractal codes are compiled in a fractal dictionary. To encode an image, FVQ approximates the image with a set of fractal code vectors from the dictionary, which is stored.

The decoder uses a standard fractal decoding algorithm since the fractal dictionary is not required by the decoder.

Keywords: Fractal Compression, Image coding, Vector Quantization

Computing Review Categories: A.1, I.3.5, I.4.2

1 Introduction

Images which are encoded and decoded with standard fractal coding methods maintain excellent image qualities as well as very high compression ratios. However, one of the major drawbacks of this method is that the encoder is computationally expensive. The fractal decoder is simple and reconstructs the fractal encoded images with little time complexity. The publications [7] and [1] can be consulted for a detailed explanation of fractal coding.
On the other hand, vector quantization (VQ) coders exhibit a much better performance in encoding time but the image quality of a VQ decoded image does not compare favourably to the results obtained with fractal coding. Another drawback of the VQ coder is that the codebook used by the encoder is also required by the decoder. Considering storage space requirements, the VQ decoder is inferior to the fractal decoder but, on the other hand, the time complexity of the VQ encoder is less than that of the fractal encoder. The publications [5] and [9] can be consulted for a detailed discussion on vector quantization.

In this paper, a novel image encoding technique utilizing the positive aspects of these two approaches is proposed. The encoding technique consists of two steps. In the first step, a fractal dictionary is set up, representing the building blocks of an arbitrary real world image. The fractal dictionary consists of the building blocks plus the fractal code which describes each of these building blocks. The fractal dictionary eliminates the need for the traditional affine-map searches which contributes to the time complexity of conventional fractal encoding methods. In the compilation of the fractal dictionary the widest possible selection of building blocks must be included. The dictionary is compiled with a vector quantization algorithm. Once the fractal dictionary has been compiled, it is modified only if a suitable building block cannot be found. In such an event, the fractal code is found in a conventional manner and is added to the dictionary.

In the second step, the image is partitioned into cells. The fractal encoder employs the fractal dictionary to approximate each partition cell with a suitable building block, and the corresponding fractal code is stored.

A conventional fractal decoder is used to decode the compressed image.

The term FVQ is used throughout this work to describe this proposed fractal vector quantization coder.

The first phase is the most time-consuming in the proposed approach, but is performed only once. The operation of the FVQ coder is depicted in figure 1.
2 Designing the FVQ coder

Fractal Dictionary

In the first phase, the fractal dictionary is set up in two steps. In the first step, an optimal codebook is constructed through vector quantization [5], and in the second stage, the codebook is fractalized [3] into the fractal dictionary.
The optimal vector codebook is designed from an initial codebook. Such an initial codebook is constructed by the use of the splitting algorithm [9]. According to this algorithm, a single, first vector is constructed by an averaging procedure (centroid calculation GERGRAY) over the entire input training set matrix. This first vector is then split recursively by adding and subtracting small fixed units. Each split vector is recorded into the initial codebook. This is done until a stopping criteria, such as a specific number of vectors, is reached. The initial codebook must be optimized so that its vectors represent real world building blocks. (A building block is a small $n \times n$ matrix which can also be written as a $1 \times m$ vector.)

The initial codebook is optimised through the LBG algorithm [9]. According to this algorithm, the codebook is run iteratively through two steps. In the first step the nearest neighbor algorithm (NN) is used to define a unique partition. In the second step, the centroid condition is used to calculate an optimal code vector for each given partition [5].

In each iteration, a new codebook is built from the previous codebook. The NN condition insists that a partition cell $R_m$ consists of all those points $x$ which have less distortion when reproduced with code vector $y_i$ than with any other code vector. The centroid condition is used to find the successive codebook which is the best reproduction for the partition cell designed in the previous step.

The codebook is considered optimal when the partial distortion becomes very small or is below a convenient (preset) threshold value.

The optimal codebook consists of code vectors, which are also considered the real world building blocks. In the second stage of constructing the fractal dictionary, the optimal codebook is subject to a fractal coding algorithm and the code vectors are fractalized.

The outcome of the fractalization process produce a dictionary which has a number of code vectors (the same is in the optimal codebook) plus the affine transformation for each of the code vectors.

The fractalization process which is based on the work of Jacquin [7] consists in principle of the following steps:

Initially an input matrix (usually an image) is segmented into non-overlapping partition cells, $R_i (0 \leq i \leq N)$, with sizes determined by the code vectors. These are called the range cells. A number of domain cells, $D_j (0 \leq j \leq M_i)$ are then generated by shifting a window of twice the range cell size across the input matrix. If a pixel spacing of 1 pixel is used in both the horizontal and vertical directions a very large number of domain cells are thus generated. On the one hand this is necessary to have a wide selection to compare range-to-domain cells to find the best domain cell match for a particular range cell. On the other hand, a large number of domain cells complicates the search time and results in slow encoding times.
In the next step, an affine transformation $\tau_i : D_j \rightarrow R_i$ is defined for each range cell $R_i$. When the affine transform is applied to an image, a spatial transformation, $\xi_i : D_j \rightarrow R_i$, determines how a domain is mapped to a range. This transformation reflects the size of the cell and an isometry of the transformation. A massic transformation, $\gamma_i : \xi_i(D_j) \rightarrow R_i$, controls the contrast and brightness of the transformation, described by $z$. The affine transformation is written as $\tau_i(\rho_j) = \lambda_i(\alpha_i, \xi(\rho_j) + \zeta_i)$, where $\alpha_i, \zeta_i$ refers to the contrast (offset) and brightness (scaling) settings, while $\xi(\rho_j)$ represents the grey value at $\rho_j$ and $\lambda_i$ indicates the isometry. In practice, $\rho_j$ represents all the grey scale values of a particular domain cell $D_j$.

To find the contracted domain cell $\xi(D_j)$ that best fits a given range cell $R_i$, least squares regression method [6] is used. In this stage, an $R_i$ scans through all the domains, $D_j$, to find a particular $D_j$ which minimizes $d_{MSE}(R_i, \tau_i(D_j))$. The transformation with the minimum distortion is selected and stored against the input code vector (range cell) in the fractal dictionary.

**FVQ Encoder**

The encoding implementation is based on a standard VQ procedure. According to the standard VQ procedure [5] an input image is segmented into input vectors having the same dimension as the code vectors of the fractal dictionary. For each input vector, an equivalent vector from the dictionary is selected, by using some selection criterion, such as finding the vector $y_{opt}$ from the dictionary for which $d(x, y_i)$ has a minimum value, with $d$ some distance function.

If this kind of selection criteria is used, it could happen that the match between the input vector and the dictionary vector is not visually satisfactory. Hence the algorithm was improved by pre-determining a tolerance that has to be met. If the tolerance is not met, the input vector is encoded with a standard fractal encoding method. In this case, the fractal dictionary is also improved by appending the input vector and its fractal code to the dictionary. The results discussed below show that the time complexity of this improved FVQ is slightly higher than the FVQ, but the visual appearance of the result is better.

**FVQ Decoder**

During the encoding step, the fractal code of the best matched code vector is recorded. The decoding step follows the standard decoding algorithm which reads the affine transformations and applies each one in turn to a random initial image. The process is iterated until an image of acceptable quality is constructed. After reconstruction of the image, the blocky artifacts visible between cell boundaries are removed by a smoothing algorithm. Grey level smoothing [2] [4] [8] is an operation which is used to remove high and low peaks usually indicating noise in images. Smoothing algorithms are usually applied to large blocks (8x8) because averaging of small (4x4) blocks may degrade the quality [10].
<table>
<thead>
<tr>
<th>Coder Performances</th>
<th>Fractal</th>
<th>VQ</th>
<th>Std. FVQ</th>
<th>Imp. FVQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition cell size</td>
<td>4x4</td>
<td>4x4</td>
<td>4x4</td>
<td>4x4</td>
</tr>
<tr>
<td>Preprocessing (seconds)</td>
<td>0</td>
<td>451</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>Encoding time (seconds)</td>
<td>25176.9</td>
<td>14</td>
<td>14</td>
<td>.93</td>
</tr>
<tr>
<td>Compression ratio</td>
<td>12.8:1</td>
<td>8:1</td>
<td>12.8:1</td>
<td>12.8:1</td>
</tr>
<tr>
<td>PSNR (dB)</td>
<td>34.8</td>
<td>30.11</td>
<td>27.70</td>
<td>31.93</td>
</tr>
<tr>
<td>Bit rate (bpp)</td>
<td>0.65</td>
<td>1</td>
<td>0.65</td>
<td>0.65</td>
</tr>
<tr>
<td>Decoding time (seconds)</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Storage space (bytes)</td>
<td>20480</td>
<td>32768</td>
<td>20480</td>
<td>20480</td>
</tr>
<tr>
<td>Extra decoder storage (bytes)</td>
<td>0</td>
<td>65536</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1. Comparison of Fractal, VQ and FVQ coders

3 FVQ coder experimental results

The architecture used in the experiments was a 133 MHz Pentium with a single Intel processor with 256 cache onboard and 44 megabyte memory.

In table 1, we present the results of a single experiment.

This table is typical of all experiments conducted, and clearly shows that the FVQ gives a better compression ratio, memory storage and encoding times when compared with VQ. The quality performance of the FVQ coder (as measured by the PSNR) is roughly of the same order as that of the VQ coder. The coding delay of the FVQ is a vast improvement over that of the standard fractal coder.

4 Conclusion

The new coder not only appears to have the same outstanding features as its two base coders, but also appears to have prevailed over their drawbacks. The FVQ technique has the additional advantage of flexibility, in the sense that the fractal dictionary is not stagnant. The algorithm is adaptive because it allows the tailoring of the fractal dictionary when it proves to be inadequate.

References


