A Hybrid Fractal Encoding technique for Image Compression

E Cloete
Department of Computer Science & Information Systems
UNISA, South Africa
cloete@alpha.unisa.ac.za

LM Venter
Department of Computer Science & Information Systems
PU for CHE, South Africa
rkwlmv@puknet.puk.ac.za

28 November, 1998

ABSTRACT

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

Our proposed approach is a multistep method. VQ is used to construct a set of primitives[1] representing small fractions of arbitary real world images. The affine transformation for each primitive is found through fractal encoding, creating a set of fractal vectors. These primitives with their corresponding fractal vectors form the *fractal dictionary*. Once the dictionary is compiled, this step is not repeated. The dictionary is used in a VQ encoding approach to approximate arbitrary input images to produce high compression ratios in acceptable encoding times. The achieved image quality, obtained after the decoding step, is also of high standard.

1 Introduction

Images which are encoded and decoded with standard fractal coding methods maintain good image qualities together with 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. Detailed explanations of fractal coding can be found in publications such as [2, 7, 13].

Another image coder, vector quantization (VQ), exhibits 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 in an integral part of both the encoder and decoder. This implies that the VQ decoder is inferior to the fractal decoder when storage space (a primary reason for coding images) is considered. The publications such as [9, 17] can be consulted for detailed discussions on vector quantization.

In this paper, a novel hybrid image coding technique utilizing the positive aspects of these two approaches is proposed. Some researchers [10, 13, 14, 21] have pointed out the similarity between variants of the VQ coder and the fractal coder, and some have also suggested a combination of the these coders. Hamzaoui et al. [12] implemented this idea when they merged the fractal and VQ coders as an integral part of a new hybrid coder to reduce the search for the best fractal code. Similar to our method (and conventional fractal coding methods), their encoded image consists of a set of precomputed affine transformations. In their coder cluster centers are precomputed according to the mean-removed shape-gain (MRSG) VQ method [9] by grouping vectors around their corresponding nearest neighbors. The

[1] In this context, a primitive is considered a most basic building block.

set of cluster centres forms the VQ codebook from which a suitable match for each range block is defined through an affine transformation. If the resulting distortion is satisfying, the range block is encoded by the affine transformation of the nearest cluster center. However, if the least square approximation yields an unacceptable high distortion, the range block is encoded by standard fractal coding. This way the search is reduced considerably, yet almost full search fidelity is maintained. A conventional fractal decoder reconstructs the original image.

In our method, we precompute not only a codebook but also the affine transformation of each codebook vector. The VQ codebook is transformed into what we call a fractal dictionary consisting of fractal vectors. Each fractal vector is constituted of a *primitive* part and a *fractal code* part. Our encoder uses the VQ method of finding the best approximation for a range block by comparing the range blocks to the primitives in the fractal dictionary. If the least square distorion is sufficiently small, the fractal code of the matching primitive is recorded. However, for unsatisfying distortions, the conventional fractal coder is employed to find a better match, after which the primitive dictionary is updated with the new primitive (contracted domain block) and fractal code (affine transformation).

Many researches [1, 15, 18, 23, 6, 22, 20] (and many more) have already suggested and established methods to improve the image qualities and search schemes of both coders. However, we restricted ourselves to basic and uncomplicated implementations of the base methods in order to introduce and establish the FVQ coder. This was done because results from simple implementations are more easily analysed than results from optimised methods.

2 Background

2.1 Vector quantization

In standard vector quatization (VQ), a vector quantizer codebook is precompiled as a set of primitives,[2] say $\{y_j\}$, which can be used to recreate an arbitrary real world image. In the encoding step, the input image is segmented into a set of input partitions, say $\{x_i\}$. A search procedure is then conducted for each x_i to find the best matching y_j from the VQ codebook. Compression is achieved when only the index, j, of the matching primitive is stored. The publications [9] and [17] can be consulted for a detailed discussion on vector quantization.

[2] These sets of primitives are usually rather small - from about 512 and up.

2.2 Fractal coding

In standard fractal coding, an imput image is segmented into a set of input partitions, say $\{x_i\}$. A copy of the input image is also segmented into analogous (but larger) partitions, say $\{y_j\}$. The difference between the two sets is that the first set consists of non-overlapping cells meant to be approximated by cells from the second set. The analogous set is comprised of overlapping partition cells in all possible sizes larger than the cells from the first set so that contraction for an analogous-to-input cell is possible. The reason for overlapping the second set of cells is to create a wide selection of cells for comparisons.

After segmentation, an intensive search is conducted in which each input partition, x_i, is matched up with every possible orientation from all the cells in the analogous set. The best match is recorded. This implies that for only one input-to-analogous comparison, several isometry operations are considered to include all possible orientations of the two cells in comparison. Remember that the analogous set could potentially be very large, which implies that the number of operations results in a lengthy search for the best analogous candidate.

During this search, each comparison operation can be described in the form of an affine transformation which is momentarily stored. When the best match is found, the corresponding affine transformation is recorded as the fractal for the particular input cell. [13], [2] or [5] can be consulted for a detailed explanation of fractal coding.

Standard fractal coding methods rise above many other image coding techniques in the sense that it maintains high image quality after decoding but presents high compression ratios during encoding. Also, the fractal decoder is simple and fast. One drawback of standard fractal coding is that it is computationally very expensive and hence time-consuming.

Vector quantization on the other hand, exhibits a much better performance in encoding times but the image quality of a VQ decoded image does not compare favourably to the results obtained with fractal coding. Another deterrent of this coding method is the additional storage requirements imposed by the VQ codebook since it is required by both the encoding and decoding procedures.

In conclusion, the VQ decoder is inferior to the fractal decoder in both staorage space requirements and image quality but, on the other hand, the time complexity of the VQ encoder is less than that of the fractal encoder.

2.3 Fractal vector quantization coder

Fractal vector quantization (FVQ) is a novel image coding technique utilizing the positive aspects of the two coding approaches mentioned above. It relies on the fundamentals of fractal coding and vector quantization. The coding is done in three steps, namely the fractal dictionary construction,

the image encoding and the image decoding.

Fractal dictionary construction

The objective in using a fractal dictionary is to reduce the time complexities associated with traditional fractal coding methods. In order to achieve this goal, the fractal dictionary is compiled once, and updated periodically.

The fractal dictionary contains fractal records consisting of two components: a primitive and an affine transformation. In the fractal dictionary construction process, (see the details below) a primitive dictionary is compiled using a vector quantization algorithm. This primitive dictionary is similar to the VQ codebook in the sense that it consists of a relative small number of primitives which can be used to approximate arbitrary digital images. The fractal dictionary consists of these primitive together with a fractal (affine transformation) that describes the primitive.

The final fractal dictionary is subject to modification during the encoding process of an arbitrary image only if a suitable fractal vector (in the comparison between the primitives and a particular input cell) cannot be found. In such an event, the affine transformation of the input partition is sought in a conventional manner. A new fractal vector is compiled by using the input partition as primitive and the newly found affine transformation as companion. The new fractal vector is appended to the fractal dictionary. (In our experiments, this did not happen often which is also clearly seen in the slight time increase from working only with the dictionary to improving the dictionary during encoding.)

The encoding process

In the second step, an input image (to be encoded), is segmented into input partition cells. The FVQ encoder uses the fractal dictionary to approximate each partition cell with a suitable fractal vector. This is achieved by a comparison of a partition cell to the primitive part of each fractal vector. Once an acceptable partition-to-primitive match is found, the affine transformation of the particular fractal code is stored. In this way, the time-consuming exhaustive searches of conventional fractal coding is avoided.

The decoding process

In the final step, a conventional fractal decoder is used to decode the compressed image.

The operation of the FVQ coder is depicted in diagram 1.

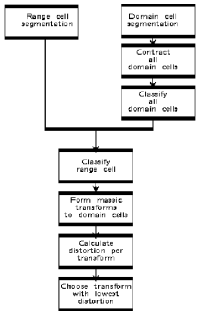

Diagram 1: Operation of the FVQ coder

3 Designing the FVQ coder

In this section, the theoretical details of the design of the FVQ coder are explained.

3.1 Fractal Dictionary

The creation of the fractal dictionary proceeds in two phases.

Phase 1: Creating the primitive dictionary

A primitive dictionary is designed from an initial codebook by using LINDE's [17] *splitting algorithm*. This initial codebook includes a set of vectors, $\{v_i\}$ which must be optimized into the primitives $\{p_i\}$ of the primitive dictionary.

The training set[3] $\{t_i\}$ serves as input to the splitting algorithm. The first vector of the codebook is generated by averaging the entire training set ($v_1 = \frac{1}{n}\sum_i^n t_i$ with n the number of training vectors). Now an iterative process is started, in which each vector v_i in the codebook is split into two new vectors $v_i + \epsilon$ and $v_i - \epsilon$, with ϵ a fixed perturbation. (In our experiments, we found that a value of $\epsilon = 0.01$ worked well in all cases.) Each split vector is recorded in the initial codebook. This is done until a stopping criteria, such as a specific number of vectors, is reached.

[3]In VQ, a training set refers to a finite collection of sample vectors generated from a source distribution in order to represent the characteristics of the source.

The initial codebook must be optimized to be able to represent all (or at least as many as feasible) possible fractions of arbitrary real world images. The LBG algorithm [17], which runs the codebook iteratively through two steps, is used to optimize the initial codebook into a primitive dictionary.

In each iteration, a new codebook is built from the previous codebook. In the first step, the code vector v_i nearest neighbor condition, which insists that a partition cell x_i consists of all those points $\mathbf{x}$ which have less distortion when reproduced with code vector $\mathbf{v}_i$ 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 (to be used as primitive dictionary) when the partial distortion[4] is below a convenient (preset) threshold value.

The primitive dictionary consists of primitives, p_i, which are considered to represent different fractions of real world images. (At this stage the set $\{p_i\}$ is the same as the set $\{v_i\}$.)

Phase 2: From primitive dictionary to fractal dictionary

In the next stage, the primitive dictionary is subject to a standard fractal coding algorthm where the primitives are *fractalized*. The outcome of the fractalization process produces the fractal vectors.

The fractalization process which is based on the work of Jacquin [13] consists in principle of the following steps:

The primitives, $p_i (0 \leq i \leq N)$, of the primitive codebook are the input cells to be subject to the rest of the algorithm. (The primitive dictionary are reformatted to represent an input matrix so that segmentation can occur naturally.)

A number of analogous cells, a_j $(0 \leq j \leq M,)$ are generated by shifting a window of twice[5] the input primitive 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 analogous cells are generated. On the one hand this is necessary to have a wide selection to compare input-to-analogous cells to find the best representative for a particular input primitive. On the other hand, a large number of analogous cells complicates the search time and results in slow encoding times.

After segmentation, an affine transformation $\tau_i : a_j \rightarrow p_i$ is defined for each input primitive. When the affine transform is applied to the input matrix, a spatial transformation, $\xi_i : a_j \rightarrow p_i$, determines how a analogous cell is mapped to an input primitive.

This transformation reflects the size of the cell and the isometry of the transformation. A massic transformation, $\gamma_i : \xi_i(a_j) \rightarrow p_i$, controls the

---

[4]The quality degradation of a decoded image is called the *distortion* incurred by the coder, and is measurable by a suitable *distortion metric*

[5]It may be any size bigger than the input primitive, twice is convenient.

contrast and brightness of the transformation. The affine transformation is written as $\tau_i(\rho_j) = \iota_i(\alpha_i.\xi(\rho_j) + \zeta_i)$, where α_i, ζ_i refers to the contrast (offset) and brightness (scaling) settings, while $\xi(\rho_j)$ represents the grey value at ρ_j and ι_i indicates the isometry. In practice, ρ_j represents all the grey scale values of a particular analogous cell a_j.

To find the contracted analogous cell $\xi(a_j)$ that best fits a given input primitive cell p_i, least squares regression method [11] is used. In this stage, an p_i scans through all the analogous cells to find a particular a_j which minimizes $d_{MSE}(p_i, \tau_i(a_j))$. The transformation with the minimum distortion is selected and the fractal vector, $f_i(p_i, \tau_i)$, is compiled by the input primitive and the corresponding affine transformation.

3.2 FVQ Encoder

The encoding implementation is based on a standard VQ encoding algorithm [9]. According to this algorithm an input image is segmented into input partition cells $\{x_i\}$ having the same dimension as the primitives of the fractal dictionary. For each cell, x_i, a primitive p_j from the dictionary is selected, by using some selection criterion, such as finding the primitive p_{opt} for which $d(x_i, p_j)$ has a minimum value, with d some distance function such as the mean squared error function.

If this kind of selection criteria is used, it could happen that the match between a particular partition and the primitive is not visually satisfactory. To overcome this visual annoyance, the algorithm was improved by predetermining a tolerance that has to be met. If the tolerance is not met, the standard fractal encoder is used to find the affine transformation for the particular input partition cell. A new fractal vector is compiled by using the input partition cell as primitive and the matching affine transformation. The fractal dictionary is improved by appending the new fractal vector to the dictionary. The results shown below shows that the time complexity of this improved FVQ is slightly higher than the FVQ, but the visual appearance of the result is better.

3.3 FVQ Decoder

During the encoding step, the affine transformation of the best matched primitive is recorded. The decoding step follows the standard fractal decoding algorithm which reads the recorded 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 [4] [8] [16] 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 (8×8) because averaging of small (4×4) blocks may degrade the quality [19].

4 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.

The error images shown in figurers xx to yy clearly illustrates that the achievable image quality of FVQ is similar to that of standard fractal coding in a fraction of the encoding time.

5 Conclusion and Future research

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.

Barthel et al. [3] consider a unification between the fractal coder and a transform coder in which they the qualities of the transform coder to reduce the blocky artifacts which is present in the fractal coder at very high bit rates and also introduces a fast codebook search scheme to select domain blocks efficiently for range block matches. In the same way, many other improvements of the base methods have already been suggested and well implemented. A sound research direction would be to introduce improved fractal encoders to design an optimal fractal dictionary. Introducing *adaptive* or *variable rate* vector quantizers would improve the quantizer performance in the construction of an optimal codebook as well as the encoding of an image. The introduction of improved algorithms should be done in conjunction with algorithm complexity, storage requirements and acceptable image quality.

FIGURE 1. Standard Fractal encoding

FIGURE 2. Standard Vector Quantization

FIGURE 3. Standard FVQ

FIGURE 4. Improved FVQ

FIGURE 5. Standard Fractal encoding

FIGURE 6. Standard Vector Quantization

FIGURE 7. Standard FVQ

FIGURE 8. Improved FVQ

Coder Performances				
	Fractal	*VQ*	*Std. FVQ*	*Imp. FVQ*
Partition cell size	4×4	4×4	4×4	4×4
Preprocessing (seconds)	0	451	500	500
Encoding time (seconds)	25176.9	14	14	93
Compression ratio	12.8:1	8:1	12.8:1	12.8:1
PSNR (dB)	34.8	30.11	27.70	31.93
Bit rate (bpp)	0.65	1	0.65	0.65
Decoding time (seconds)	2	2	2	2
Storage space (bytes)	20480	32768	20480	20480
Extra decoder storage (bytes)	0	65536	0	0

TABLE 1. Comparison of Fractal, VQ and FVQ coders

6 References

[1] B. Bani-Eqbal. Speeding up fractal image compression. In M. Rabbani, E. J. Delp, and S. A. Rajala, editors, *Still Image Compression*, volume 2418 of *SPIE*, pages 67–74, San Jose, CA, USA, February 1995.

[2] M. F. Barnsley and L. P. Hurd. *Fractal Image Compression.* A K Peters, Wellesley, Massachusetts, 1993.

[3] K. U. Barthel, J. Schüttemeyer, T. Voyé, and P. Noll. A new image coding technique unifying fractal and transform coding. In *IEEE International Conference on Image Processing*, volume III of *ICIP*, pages 112–116, Austin, Texas, USA, November 1994.

[4] H. Bassman and P. W. Besslich. *Ad Oculos, Digital Image Processing.* International Thompson Publishing, Sheffield, 1995.

[5] E. Cloete and L. M. Venter. Fractal image compression. *South African Science Journal*, 1998. August 1998.

[6] W. H. Equitz. Fast algorithms for VQ picture coding. *An International Conference on Acoustics, Speech, and Signal Processing*, pages 725–728, 1987.

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

[8] Y. Fisher, S. Perkins, A. Walker, and E. Wolfart. Spatial filters. *WWW*, 1994. http://q127-3.coventry.ac.uk/hipr/csmooth.html.

[9] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression.* Kluwer Academic Publishers, Boston, 1992.

[10] M. Gharavi-Alkhansari and T. S. Huang. Fractal-based techniques for a generalized image coding method. In *IEEE International Conference on Image Processing*, volume III of *ICIP*, pages 112–116, Austin, Texas, USA, November 1994.

[11] P. E. Gill, W. Murray, and M. H. Wright. *Numerical Linear Algebra and optimization*. Addison-Wesley Publishing Company, Redwood City, California, USA, 1991.

[12] R. Hamzaoui, M. Müller, and D. Saupe. Vq-enhanced fractal image compression. In *IEEE International Conference on Image Processing*, ICIP'96, Lausanne, September 1996. Also available by anonymous ftp in ftp.informatik.uni-freiburg.de in documents/papers/fractal/Hamz96b.ps.gz.

[13] A. Jacquin. Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Transactions on image processing*, 1(1):18–30, January 1992.

[14] C.-S. Kim and R. H. Park. Image coding based on fractal approximation and vector quantization. In *IEEE International Conference on Image Processing*, volume III of *ICIP*, pages 268–271, Washington, D.C., USA, October 1995.

[15] S. Lepsoy and G. E. Oien. Fast attractor image by adaptive codebook clustering. Fractal Image Compression - Theory and Application, pages 177–198, New York, 1995. Springer-Verlag.

[16] R. Lewis. *Practical Digital Image Processing*. Ellis Horwood Limited, West Sussex, England, 1990.

[17] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, COM-28(1):84–95, January 1980.

[18] D. M. Monro and S. J. Woolley. Fractal image compression without searching. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 5 of *ICASSP*, pages 557–560, Adelaide, Australia, April 1994.

[19] M. Nelson. *The Data Compression Book*. M & T Books, New York, 1996.

[20] B. Ramamurthi and A. Gersho. Classified vector quantization of images. *IEEE Transaction Communications*, COM-34, Nov 1986.

[21] T. E. Ramstad and S. Lepsoy. Block-based attractor coding: Potential and comparison to vector quantization. Conference Record of the 27th Asilomar Conference on Signals, Systems and Computers, pages 1504–1508, 1993.

[22] D. Saupe. Breaking the time complexity of fractal image compression. Technical Report 53, Institut für Informatik, Freiburg, Germany, 1994.

[23] D. Saupe. Accelerating fractal image compression by multi-dimensional nearest neigbor search. In J. A. Storer and M. Cohn, editors, *IEEE Data Compression Conference*, DCC, pages 222–231, UT, USA, March 1995. Snowbird.