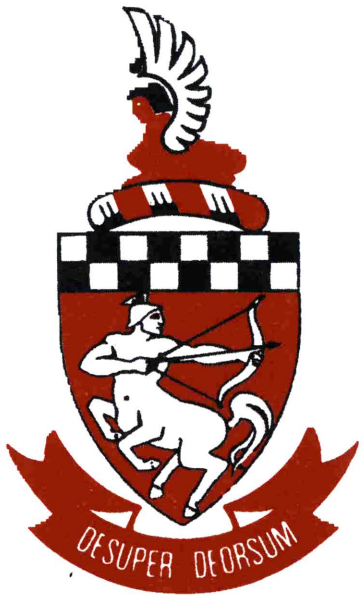
South African Computer Journal

Number 25
August 2000

Suid-Afrikaanse Rekenaar-Tydskrif

Nommer 25
Augustus 2000



The South African Computer Journal
An official publication of the Computer Society of South Africa and the South African Institute of Computer Scientists

Die Suid-Afrikaanse Rekenaartydskrif
'n Amptelike publikasie van die Rekenaarvereniging van Suid-Afrika en die Suid-Afrikaanse Instituut vir Rekenaarwetenskaplikes

World-Wide Web: `http://www.cs.up.ac.za/sacj/`

Subscriptions

	Annual	Single copy
Southern Africa	R80.00	R40.00
Elsewhere	US$40.00	US$20.00

An additional US$15 per year is charged for airmail outside Southern Africa

to be sent to:

Computer Society of South Africa
Box 1714, Halfway House, 1685
Phone: +27 (11) 315-1319 Fax: +27 (11) 315-2276

In Memoriam:
Stef Postma

South Africa has lost one of her most colourful and eminent computer scientists. Professor Stef Postma passed away peacefully in his sleep on May 5, 2000 after a short illness. He will be remembered for his forthright views and total integrity. Never a man to shy from controversy, he always debated his position with vigour, displaying his extensive vocabulary at every opportunity.

Those who knew him mourn the loss of a very good friend.

Stef was born on August 10, 1938 in Graaff-Reinet and matriculated from Höerskool Linden in Johannesburg. He majored in geology and mathematics at the University of the Witwatersrand and graduated with honours in mathematics from that university. Stef devoted much of his life to promoting computer science as a science and to this end spent a lot of energy and time defining syllabi for undergraduate and honours courses at our universities. He was the prime mover in creating the South African Institute of Computer Scientists and Information Technologists (SAICSIT) in 1982, providing a professional body to represent the interests of local computer scientists. He was also instrumental in establishing Quaestiones Informaticae (now the South African Computer Journal) which afforded South African computer scientists the opportunity to publish papers locally in a refereed journal.

-Doug Laing

Links2Go "Computer Science Journals" Award

The SACJ production editing team received the following unsolicited e-mail:

The page at <http://www.cs.up.ac.za/sacj/>, was selected as a Links2Go "Key Resource" in the Computer Science Journals topic, at <http://www.links2go.com/topic/Computer_Science_Journals>.

How your page was selected:

Each quarter, Links2Go samples millions of web pages to determine which pages are most heavily cited by web pages authors, such as yourself. The most popular pages are downloaded and automatically categorized by topic. At most 50 of the pages related to a topic are selected as "Key Resources." Out of 50 pages selected as Key Resources for the Computer Science Journals topic, your page ranked 19th. For topics like Music, where there are a large number of interested authors and related pages, it is harder to achieve selection as a Key Resource than for a special-interest topic, such as Quantum Physics.

The Links2Go Key Resource award differs from other awards in two important ways. First, it is objective. Most awards rely on hand selection by one or more "experts," many of whom have only looked at tens or hundreds of thousands of pages in bestowing their awards. Selection for these awards means no more than that one person, somewhere, noticed your page and liked it enough to select it. The Key Resource award, on the other hand, is based on an analysis of millions of web pages. Any group or organization who conducts a similar analysis will arrive at similar conclusions. When Links2Go says your page is a Key Resource, we mean that your page is one of the most relevant pages related to a particular topic on the web today, using an objective statistical measure applied to an extremely large data set.

Second, the Key Resource award is exclusive. We get literally hundreds of people requesting that their page be added to one or more topics per week. All of these requests are denied. The only way to get listed as a Key Resource is to achieve enough popularity for our analysis to select your pages automatically. We do not accept fees, offers of link exchanges, free advertising, or bartered livestock as inducements to add new sites to our lists. Fewer than one page in one thousand will ever be selected as a Key Resource.

Once again, congratulations on your award! Links2Go Awards awards@links2go.com

Research Article

Image coding with Fractal Vector Quantization

E Cloete[a] LM Venter[b]

[a]*Department of Computer Science and Information Systems, UNISA, Pretoria, 0001*
[b]*Department of Computer Science and Information Systems, PU for CHE, Vanderbijlpark, 1900* [a]cloete@alpha.unisa.ac.za

Abstract

In this paper we address the time complexity problem associated with fractal image coding. In partcular, we describe a new hybrid technique called Fractal Vector Quantization (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 describes the block. The real world building blocks with their affine transformations are compiled in a fractal dictionary. To encode an image, FVQ approximates the image with a set of affine transformations from the precompiled fractal dictionary. 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 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, 6, 14].

Another image coder, vector quantization (VQ), 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 drawback of the VQ coder is that the codebook forms 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. Publications such as [8, 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 [9, 14, 15, 20] 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 [11] 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 affine transformations. 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 [8] by grouping vectors around their corresponding nearest neighbours. The 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 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 further work Hamzaoui et al [12, 13] refine this research effort.

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, 16, 18, 22, 5, 21, 19] (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 The principle of the FVQ coder

The *FVQ coder* is consists of three elements namely the *fractal dictionary*, representing the building blocks of an arbitrary real world image, the *encoder*, encoding arbitrary images, and the *decoder*, reconstructing an estimation of the original image.

The design of the *fractal dictionary* is the most time-consuming algorithm in the proposed approach. Different to the virtual codebook of conventional fractal coding the fractal dictionary is designed only once as a pre-processing step. The fractal dictionary eliminates the need for the traditional affine map searches which contributes to the time complexity of conventional fractal encoding methods.

To accurately describe any arbitrary image, it is important that the widest possible selection of building blocks must be included in the dictionary which is compiled with a VQ algorithm. Once the fractal dictionary has been compiled the FVQ encoder is used to encode arbitrary image. This coder is improved by expanding the fractal dictionary during encoding through a learning subsystem to incorporate new building blocks.

In the design of the *encoder*, the image is partitioned into into range block cells. The FVQ encoder employs the fractal dictionary to approximate each range cell with a suitable primitive from the fractal dictionary, and the corresponding fractal code is recorded.

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

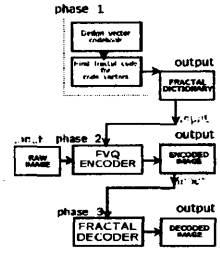


Figure 1: Operation of the FVQ coder

The *FVQ decoder* follows the standard fractal 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, some of the blocky artifacts visible between cell boundaries are removed by a smoothing [7] algorithm.

2.1 Fractal dictionary design

A fractal dictionary is constructed through a number of steps. Firstly, an initial VQ codebook is designed by the use of the splitting algorithm [17] in which a single, first vector is constructed by an averaging procedure (centroid calculation [8]) over an entire input training set matrix[1]. This first vector is then split recursively by adding and subtracting small fixed units. Each split vector is recorded into the initial VQ 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.

The initial codebook is optimised through the LBG algorithm [17] which runs iteratively through two steps. In the first step the nearest neighbour algorithm is used to define a unique partition. The nearest neighbour condition insists that a partition cell consists of all those points which have less distortion when reproduced with a particular code vector than with any other code vector. In the second step, the centroid condition is used to calculate an optimal code vector for the given partitions. In each iteration, an improved VQ codebook is built from the previous codebook. The codebook is considered optimal when the partial distortion becomes very small or is below a convenient threshold value. The optimal codebook consists of code vectors which are 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 produces a dictionary which has a number of primitives (the same as the code vectors in the optimal codebook) plus the fractal code (affine transformation) of each primitive.

The fractalization process which is based on the work of Jacquin [14] consists in principle of the following steps: Initially the fractal dictionary (written in image format) is segmented into non-overlapping range cells, $R_i (0 \leq i \leq N)$, with sizes determined by the code vectors. A number of domain cells, D_j $(0 \leq j \leq M,)$ are then generated by shifting a window of twice the range cell size across the input matrix.

In the next step, an affine transformation τ_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 $\tau_i(\rho_j) = \iota_i(\alpha_i.\xi(\rho_j) + \zeta_i)$, where α_i and ζ_i refers to the con-

[1] In VQ, a training set refers to is a finite collection of sample vectors generated from a source distribution in order to represent the statistics of the source with a finite data set.

trast (offset) and brightness (scaling) settings, while $\xi(\rho_j)$ represents the grey value at ρ_j and ι_i indicates the isometry.

Least squares regression [10] is used to find the contracted domain cell $\xi(D_j)$ that best fits a given range cell R_i. 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 recorded with its primitive (range cell) in the fractal dictionary.

2.2 Encoder design

The encoding implementation is based on a standard VQ procedure. According to the standard VQ procedure [8] an input image is segmented into input vectors having the same dimension as the primitives of the fractal dictionary. For each input vector, an equivalent primitive from the dictionary is selected, by using some selection criterion, such as finding the primitive y_{opt} from the dictionary for which $d(x, y_i)$ has a minimum value, with d the least squares 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. The FVQ coder is then improved by the inclusion of a predetermined 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 time complexity of this improved FVQ coder is slightly higher than an FVQ coder without the adaptive dictionary FVQ, but the visual appearance of the result is much better.

3 FVQ CODING 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. Coding experiments carried out in this work were limited to the use of grey scale input images. In comparing the FVQ, fractal and VQ coders, we show the results off the 512×512 image of Lena using a 4×4 fixed-sized block partitioning. (The results presented here are typical of some 257 experiments conducted, and we present the result of a single expleriment. Full details of this research work are available in [4].)

Table 1 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 also better than that of the VQ coder.

The coding delay of the FVQ is a vast improvement over that of the standard fractal coder.

From the graphs given in figures 2 and 3 the FVQ coder seems to be very attractive because the encoding time is not excessive while the PSNR is high. The VQ and FVQ coders perform well in encoding times and are attractive coders in applications where the perceptual image quality does not play a major role. The fractal coder yields the best image quality, but is hopelessly inadequate in a real-time environment. The result for the FVQ coder becomes more optimal as the fractal dictionary is improved. Comparative evaluation of the FVQ encoder shows that the time complexities of standard fractal encoding are reduced when encoding multiple images. This is also seen in the exponential shape of the curve for the fractal coder. The same is true for the FVQ encoder with an adaptive dictionary (compare to the FVQ with a fixed dictionary). To retain objectivity in the comparison, all times recorded here were subject to algorithms using full-search techniques.

Although it seems that the FVQ encoder has a compression ratio similar to that of the fractal encoder, this is not, in fact, true. A 25% compression ratio is already obtained when the VQ codebook is fractalized, which introduces a certain amount of error. Considering file storage, the FVQ coder pair performs roughly as well as the fractal encoder, but outperforms the VQ encoder by about 32%. The significant advantage of this method is that it retains the fast decoding times of the fractal decoder.

4 Conclusion and future research

The FVQ 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 since 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.

References

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

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

Table 1: Comparison of Fractal, VQ and FVQ coders

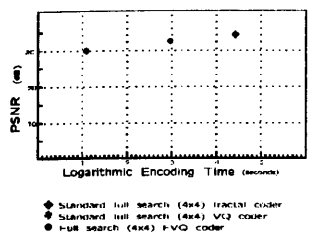


Figure 2: Comparison of visual qualities of decoded images through Fractal, VQ and FVQ coders

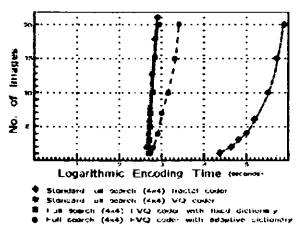


Figure 3: Comparing algorithm complexities of the Fractal, VQ and FVQ coders.

[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*, pp. 112–116, Austin, Texas, USA, (November 1994).

[4] E Cloete. 'Using vector quantization for fractal coding'. *PhD Dissertation*, (May 1998).

[5] W H Equitz. 'Fast algorithms for VQ picture coding'. *An International Conference on Acoustics, Speech, and Signal Processing*, pp. 725–728, (1987).

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

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

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

[9] 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*, pp. 112–116, Austin, Texas, USA, (November 1994).

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

[11] R Hamzaoui, M Müller, and D Saupe. 'Vq-enhanced fractal image compression'. In *IEEE Int. Conf. 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.

[12] R Hamzaoui and D Saupe. 'Combining fractal image compression and vector quantization'.

Institut für Informatik, Universität Leipzig, Germany, (August 1998). Available by anonymous ftp in ftp.informatik.uni-freiburg.de in documents/papers/fractal/HamzSa98.ps.gz.

[13] R Hamzaoui, D Saupe, and K Barthel. 'Vq-encoding of luminance parameters in fractal coding schemes'. To appear in Proc.ICASSP-97, Munich, Germany, (April 1997). Available by anonymous ftp in ftp.informatik.uni-freiburg.de in documents/papers/fractal/Hamz96b.ps.gz.

[14] 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).

[15] 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*, pp. 268–271, Washington, D.C., USA, (October 1995).

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

[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*, pp. 557–560, Adelaide, Australia, (April 1994).

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

[20] 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, pp. 1504–1508, (1993).

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

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

South African Computer Journal

Number 25, August 2000
ISSN 1015-7999

Suid-Afrikaanse Rekenaar-tydskrif

Nommer 25, Augustus 2000
ISSN 1015-7999

Notes for Contributors

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems, as well as shorter technical research notes. However, non-refereed review and exploratory articles of interest to the journal's readers will be considered for publication under sections marked as Communications of Viewpoints. While English is the preferred language of the journal, papers in Afrikaans will also be accepted. Typed manuscripts for review should be submitted in triplicate to the editor.

Form of Manuscript
Manuscripts for *review* should be prepared according to the following guidelines:

- Use wide margins and $1\frac{1}{2}$ or double spacing.
- The first page should include:
- the title (as brief as possible)
- the author's initials and surname
- the author's affiliation and address
- an abstract of less than 200 words
- an appropriate keyword list
- a list of relevant Computing Review Categories
- Tables and figures should be numbered and titled.
- References should be listed at the end of the text in alphabetic order of the (first) author's surname, and should be cited in the text according to the Harvard. References should also be according to the Harvard method.

Manuscripts accepted for publication should comply with guidelines as set out on the SACJ web page,

`http://www.cs.up.ac.za/sacj`

which gives a number of examples.

SACJ is produced using the LaTeX document preparation system, in particular LaTeX 2_ε. Previous versions were produced using a style file for a much older version of LaTeX, which is no longer supported. Please see the web site for further information on how to produce manuscripts which have been accepted for publication.

Authors of accepted publications will be required to sign a copyright transfer form.

Charges
Charges per final page will be levied on papers accepted for publication. They will be scaled to reflect typesetting, reproduction and other costs. Currently, the minimum rate is R30.00 per final page for contributions which require no further attention. The maximum is R120.00, prices inclusive of VAT.

These charges may be waived upon request of the author and the discretion of the editor.

Proofs
Proofs of accepted papers may be sent to the author to ensure that typesetting is correct, and not for addition of new material or major amendments to the text. Corrected proofs should be returned to the production editor within three days.

Letters and Communications
Letters to the editor are welcomed. They should be signed, and should be limited to about 500 words. Announcements and communications of interest to the readership will be considered for publication in a separate section of the journal. Communications may also reflect minor research contributions. However, such communications will not be refereed and will not be deemed as fully-fledged publications for state subsidy purposes.

Book Reviews
Contributions in this regard will be welcomed. Views and opinions expressed in such reviews should, however, be regarded as those of the reviewer alone.

Advertisement
Placement of advertisements at R1000.00 per full page per issue and R500.00 per half page per issue will be considered. These charges exclude specialised production costs, which will be borne by the advertiser. Enquiries should be directed to the editor.

South African Computer Journal

Number 25, August 2000
ISSN 1015-7999

Suid-Afrikaanse Rekenaar-tydskrif

Nommer 25, August 2000
ISSN 1015-7999

Contents