PROGRESSIVE FRACTAL CODING

Iván Kopilović, Dietmar Saupe, Raouf Hamzaoui

Universität Leipzig, Institut für Informatik, Augustusplatz 10–11, 04109 Leipzig, Germany

ABSTRACT

Progressive coding is an important feature of compression schemes. Wavelet coders are well suited for this purpose because the wavelet coefficients can be naturally ordered according to decreasing importance. Progressive fractal coding is feasible, but it was proposed only for hybrid fractal-wavelet schemes. We introduce a progressive fractal image coder in the spatial domain. A Lagrange optimization based on rate-distortion performance estimates determines an optimal ordering of the code bits. The optimality is in the sense that the reconstruction error is monotonically decreasing and minimum at intermediate rates. The decoder recovers this ordering without side information. As a side effect, our work motivates improved bit allocation strategies for fractal coding.

1. INTRODUCTION

A progressive image code is such that a sequence of prefixes of the code can be interpreted by the decoder, yielding reconstructions whose quality increases monotonically with the length of the prefix. This is a useful property because it allows to stop decoding at an early stage if one is already satisfied with the quality of the reconstructed image or if the image is not appropriate. Most modern compression schemes such as the embedded wavelet coder of Shapiro [8] or JPEG2000 generate progressive image codes.

Standard fractal image coding [4] is a spatial domain technique. The code consists of parameters for a contractive affine transformation, which when iterated, converges to an approximation of the original image. The affine transformation is given by a partition of the image into blocks, which are approximated by downsampled, scaled, and grey-level shifted blocks of the same image. In this setting, the progressive transmission of the fractal code is a difficult task because the relative importance of the transformation parameters is not obvious. Fractal image coding can also be done in the wavelet domain [3] allowing the generation of progressive codes [2].

We first define a rate-distortion optimality criterion for the transmission of an image code. Our objective is to reorder the bit-stream such that the reconstruction error is monotonically decreasing and minimum at intermediate rates. We use Lagrange optimization to design a spatial domain progressive fractal coder that satisfies the optimality criterion. Our encoder sends the bits of the fractal code according to rate-distortion importance, and the decoder recovers this ordering by retracing the optimization procedure of the encoder without requiring any side information bits. Our approach is similar to the one used in [6] for wavelet coders.

2. PROGRESSIVE CODING: THEORY

The bit-stream of any image code contains values for parameters used by the compression method. These parameters can be represented by one or more bits. Our goal is to order the parameters so that if the corresponding bit-stream is stopped at some point, the image reconstruction based on the partial knowledge of the code is optimal for the number of bits received.

Let us denote the set of the parameters with Ω. The image code is given by a mapping $Q : \Omega \to \{0,1\}^+$, where $\{0,1\}^+$ it the set of all finite sequences of bits. For each parameter $\omega \in \Omega$ we denote the number of bits in $Q(\omega)$ by $\ell(\omega)$. For $A \subset \Omega$ we set $\ell(A) = \sum_{\omega \in A} \ell(\omega)$.

We formulate the problem of progressive coding as follows. Suppose that the bit-stream is stopped after receiving the bits corresponding to the set of parameters $A \subset \Omega$ with $m = \ell(A)$. If the decoder can identify A, along with the order of transmitted bits in A then it will know $Q(\omega)$ for all $\omega \in A$. Then the decoder must apply a bit assignment strategy $\Phi_A : \Omega \setminus A \to \{0,1\}^+$ for the remaining unknown parameters $\omega \in \Omega \setminus A$. By defining a full code Q_A as $Q_A(\omega) = Q(\omega)$ for $\omega \in A$ and $Q_A(\omega) = \Phi_A(\omega)$ otherwise, the decoder may proceed to produce a reconstruction yielding some error $E(A)$. We denote by $\mathcal{P}$ the set of all $A \subset \Omega$ for which the decoder has an assignment strategy Φ_A. We shall see in Section 4 that there exist subsets of Ω not belonging to $\mathcal{P}$.

Suppose that the bit-stream is allowed to be stopped at n successive points, where the nth point is the end of the total bit-stream. At each stage $k = 1, \ldots, n$, the received bits will correspond to a parameter set $A_k \in \mathcal{P}$, where $A_n = \Omega$. Thus we have a sequence $A_1 \subset A_2 \subset \cdots \subset A_n$ of parameter sets, which correspond to the stages of the progressive transmission.

Let $m_1 < m_2 < \cdots < m_n = \ell(\Omega)$ be a sequence of numbers defining the *rate constraints*. The code Q together with the parameter ordering $A_1 \subset A_2 \subset \cdots \subset A_n = \Omega$, where $A_k \in \mathcal{P}$ for $k = 1, \ldots, n$, is called an *optimal progressive code* with respect to the error measure E under the given rate constraints if

$$E(A_k) = \min\{E(A) \mid A \in \mathcal{P}, A \supset A_{k-1}, \ell(A) \leq m_k\}, \quad (1)$$

for all stages $k = 1, \ldots, n$ with $A_0 = \emptyset$.

An optimal progressive coder/decoder system can be obtained as follows. The encoder uses an optimization procedure to successively generate the parameter sets A_k satisfying (1) based on the bits generated in stage $k-1$. The decoder uses the same optimization procedure and mirrors the parameter subset generation of the encoder. In this way, the decoder is able to track the bit-stream and to interpret the incoming bits.

Generating the parameter subsets of the progressive code is a constrained optimization problem. We use the Lagrange multiplier method to handle the rate constraints. The Lagrange functional is defined as $L(A, \lambda) = E(A) + \lambda \ell(A)$, for $A \in \mathcal{P}$ and $\lambda \geq 0$.

Given the real parameters $\lambda_1 \geq \cdots \geq \lambda_n$, we generate an optimal progressive code with the following algorithm. We set $A_0 = \emptyset$ and define

$$A_k \in \arg\min\{L(A, \lambda_k) \mid A \in \mathcal{P}, A \supset A_{k-1}\}. \qquad (2)$$

The parameter sets $A_1 \subset \ldots \subset A_n$ generated by the above algorithm will satisfy (1) for the rate constraints $m'_k := \ell(A_k)$, $k = 1, \ldots, n$. The parameters λ_k, $k = 1, \ldots, n$, are determined by a search method such that m'_k is close to m_k.

3. FRACTAL CODING

In fractal image compression, the encoder finds a contractive image operator T whose fixed point f_T is an approximation to the original image f. The decoder constructs f_T as the limit of the sequence of iterates $\{f^{(k)}\}_{k\geq 0}$, where $f^{(k+1)} = T(f^{(k)})$ and $f^{(0)}$ is an arbitrary initial image. The operator T is given by a partition of the image support $\mathcal{I}$ into non-overlapping blocks called *ranges* and by *fractal parameters* associated to each range $R_i,\ 1 \leq i \leq n_R$, which consist of

- a block (*domain*) $D_i \subset \mathcal{I}$,
- a *scaling factor* $s_i \in]-1, 1[$,
- and an *offset* $o_i \in \mathbb{R}$,

which are selected such that the best affine approximation $T_i(R_i)$ for R_i in ℓ^2-sense [9] is

$$T_i(R_i) = s_i S_i(D_i - \mu(D_i)\mathbf{1}_{D_i}) + o_i \mathbf{1}_{R_i}, \qquad (3)$$

where S_i is an operator downsampling its argument via pixel averaging to match the range size, $\mu(D_i)$ is the mean value of the pixel intensities over D_i, and $\mathbf{1}_X$ is the block with unit intensity at every pixel of X. By construction, the vectors

$$\mathbf{b}_i^s = S_i(D_i - \mu(D_i)\mathbf{1}_{D_i}) \text{ and } \mathbf{b}_i^o = \mathbf{1}_{R_i}$$

used to approximate R_i are orthogonal. It follows from the least squares method that

$$s_i = \frac{|R_i|\langle \mathbf{b}_i^s, R_i\rangle - |R_i|^2\mu(D_i)\mu(R_i)}{|R_i|\|\mathbf{b}_i^s\|_2^2 - \mu(D_i)^2} \text{ and } o_i = \mu(R_i),$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product, $\|\cdot\|_2$ the Euclidian norm, and $|R_i|$ the number of pixels in R_i. We note that $\|\mathbf{b}_i^o\|_2^2 = |R_i|$. Thus the value of $\|\mathbf{b}_i^o\|_2^2$ depends only on the partition. The approximation error $\|T(f) - f\|_2^2 = \sum_{i=1}^{n_R} \|T_i(R_i) - R_i\|_2^2$ is called the *collage error*. It is often used as an estimation for the *reconstruction error* $\|f - f_T\|_2^2$. Note that $T(f)$ and the residual $T(f) - f$ are orthogonal by construction.

The fractal code for an image consists of bits for the image partition and bits for the quantized fractal parameters. In this paper we use a quadtree to describe the partition [4]. The offset and scaling parameters are encoded as quantized values with n_o and n_s bits respectively. The domain D_i is specified as a position in a spiral around the range block R_i (see [1, 7]). The position is encoded with variable length encoding. The number of bits allocated for the domain D_i will be denoted by n_{D_i}.

4. PROGRESSIVE FRACTAL CODING

In this section we discuss the concepts of progressive coding from Section 2 for fractal image compression. We explain the parameter set Ω, the set $\mathcal{P}$ of feasible parameter subsets $A \subset \Omega$, the bit assignment strategy Φ_A for completing a partial code, and finally two choices of the error measure E used in (1) and (2).

As described in Section 3, a fractal code consists of the binary representation of the header information including the image partitioning in n_R range blocks, and the binary representation of the list of tuples (D_i, s_i, o_i), $i = 1, \ldots, n_R$, each one consuming $n_{D_i} + n_s + n_o$ bits. We define the parameters $\omega \in \Omega$ for progressive coding as follows.

1. The header is a parameter.
2. Each bit in the binary representations of the scaling and offset numbers s_i and o_i is a parameter, except the most significant bit of each scaling factor s_i.
3. For $i = 1, \ldots, n_R$, the most significant bit of the scaling factor s_i, together with all n_{D_i} bits for the address of the corresponding domain block D_i is a parameter.

The motivation for these choices follows from the principle that the parameters should be regarded as atomic in the sense that any proper part of the binary representation of a parameter is useless for the purpose of decoding. Therefore, the header information should not be allowed to be split into several parameters; it is vital for the interpretation of the rest of the code. Also the domain block address is meaningful only as an entity; a partial address typically points to an unrelated domain block. Furthermore, the address still is useless if no information about the corresponding scaling factor is available. Thus, the minimal useful entity is the domain block address together with the most significant bit of the binary representation of the corresponding scaling factor. The lengths $l(\omega)$ of these parameters are simply given by the number of corresponding bits, i.e., 1, $1 + n_{D_i}$, and the number of bits of the header.

We now define the set $\mathcal{P}$ of feasible parameter subsets $A \subset \Omega$ so that we can derive a suitable heuristic bit assignment strategies Φ_A for complements $\Omega\backslash A$. We give this definition in the form of two rules:

1. For $A \in \mathcal{P}$, the header parameter is contained in A.
2. If A contains a parameter corresponding to a bit of the binary representation of a scaling factor s_i or of an offset o_i, then A contains all parameters corresponding to the bits of the scaling factor or the offset that are more significant.

The first rule ensures that the header is sent first in any progressive code. The second rule implies that bits for scaling factors and offsets are sent in order of significance.

For a feasible parameter set $A \in \mathcal{P}$ the natural bit assignment strategy $\Phi_A : \Omega\backslash A \to \{0,1\}^+$ is as follows. If no bits for a scaling factor s_i are in A, then set the bits of s_i such that $s_i = 0$. Thus, the domain address becomes irrelevant and it can be chosen arbitrarily. In all other cases, set to 0 all bits of s_i and of o_i that are not in A, except for the most significant ones of these, which are set to 1. In this way the scaling factors and offsets are dequantized to numbers in the center of the corresponding intervals that are prescribed by the information already contained in A.

For the error measure $E(A)$ in the optimization we consider the collage error. Assume that we have received the partial progressive code $Q(A)$. We then complete the code by $\Phi_A(\Omega\backslash A)$. After dequantization we obtain the transform parameters $s_{A,i}$ and

$o_{A,i}$ for $i = 1, \ldots, n_R$. The affine transformations defined in (3) using these parameters will be denoted by $T_{A,i}$. Recall from Section 3 that $\mathbf{b}_i^s, \mathbf{b}_i^o$, $i = 1, \ldots, n_R$ and $T(f) - f$ are orthogonal vectors. Using this fact, the collage error for the transform T_A, derived from parameters in A, can be expressed as

$$E(A) = \sum_{i=1}^{n_R} \|T_{A,i}(R_i) - R_i\|_2^2 = \sum_{i=1}^{n_R} \|T_i(R_i) - R_i\|_2^2 + \sum_{i=1}^{n_R} |s_{A,i} - s_i|^2 \|\mathbf{b}_i^s\|_2^2 + |o_{A,i} - o_i|^2 \|\mathbf{b}_i^o\|_2^2 \quad (4)$$

The first term on the right hand side is the collage error of the complete fractal code. It does not depend on A and may be precomputed (or omitted) for the optimization. With these settings a progressive fractal code with respect to collage error E can be constructed using the Lagrange multiplier method in Section 2.

However, using this scheme the decoder is not able to identify the parameters belonging to the received partial code $Q(A)$. We solve this problem as follows. At stage k the progressive encoder uses the error measure $E(A)$ and the length function $l(\omega)$ in order to derive the next parameters $A_{k+1} \backslash A_k$ of the code using (1). We modify $E(A)$ and $l(\omega)$ such that the decoder is capable of retracing the optimization of the encoder. We omit the constant term in (4) and replace $E(A)$ by the estimate

$$E'(A) = \sum_{i=1}^{n_R} \mathcal{E}(|s_{A,i} - s_i|^2)\chi_{A,i} + \mathcal{E}(|o_{A,i} - o_i|^2)\|\mathbf{b}_i^o\|_2^2, \quad (5)$$

where $\mathcal{E}$ denotes an estimation of the given scalars, and $\chi_{A,i}$ is an estimate for $\|\mathbf{b}_i^s\|_2^2$. Since the partition is received at the beginning, the value $\|\mathbf{b}_i^o\|_2^2$ is known to the decoder. With the scaling factors and offsets modeled as independent and uniformly distributed random variables we can straightforwardly compute appropriate expectations yielding the estimates $\mathcal{E}(|s_{A,i} - s_i|^2)$ and $\mathcal{E}(|o_{A,i} - o_i|^2)$.

We now discuss the estimation $\chi_{A,i}$ of $\|\mathbf{b}_i^s\|_2^2$. Recall that $\|\mathbf{b}_i^s\|_2^2$ is the square norm of the dynamic part of the downsampled domain block in the original image for the i-th range block R_i. Let f_{T_k} denote the image reconstruction of the decoder at stage k. We construct the corresponding domain block using the current reconstruction f_{T_k} in place of the original f and denote it by $\mathbf{b}_{i,k}^s$. If the domain block address (and the most significant scaling bit) are already included in A_k we set $\chi_{A,i} = \|\mathbf{b}_{i,k}^s\|_2^2$. Otherwise, $\mathbf{b}_{i,k}^s$ is zero, and we must estimate $\|\mathbf{b}_i^s\|_2^2$ by other means. The number $s_i^2\|\mathbf{b}_i^s\|_2^2$ is equal to the variance in the range block R_i of the reconstruction f_T multiplied by the number of pixels in the range. Therefore we use an estimate $\chi_{A,i}$ for $\|\mathbf{b}_i^s\|_2^2$ that is proportional to the variance of the current reconstruction f_{T_k} in a neighborhood N_i of the range block R_i, e.g., given by the pixels bordering R_i. Let $o_{i,k}$ be the offset of the range R_i at stage k and let h_k be the residual image $h_k(x) = f_{T_k}(x) - o_{i,k}$, for $x \in N_i$, and $h_k(x) = 0$ otherwise. We set $\chi_{A,i} = c_i\|h_k\|_2^2$, where the constant c_i is chosen proportional to the number of pixels in the range block R_i and inversely proportional to the number of pixels in the neighborhood N_i.

Finally, we have to estimate the length $l(\omega)$ when the parameter ω contains a domain block address, since the addresses are variable length encoded. The codeword length of the domain address was estimated with $\frac{1}{n_R}\sum_{i=1}^{n_R} n_{D_i}$ which number is known to the decoder, because it is included in the header.

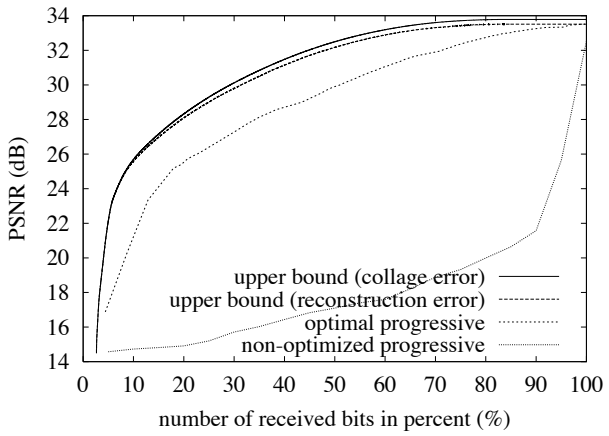

Fig. 1. Progressive coding for the 512×512 image Lenna using a quadtree fractal code at 0.5 bits per pixel. The top two curves show upper bounds for progressive fractal coding. The third curve displays the reconstruction errors for a practical progressive coder. The bottom curve shows reconstruction errors with normal order.

The computation of optimal progressive codes with respect to collage error (4) or the estimate (5) is not practical by straightforward exhaustive search in (2). The error models $E(A)$ and $E'(A)$ each consist of $2n_R$ terms (ignoring the first constant sum in (4)) which allows us to rewrite the Lagrangian $L(A, \lambda)$ as a sum of $2n_R$ corresponding terms. Each one of these terms concerns errors regarding only one range block as they depend on parameters containing bits for the scaling factor or bits for the offset. Therefore each of the $2n_R$ Lagrangian terms may be minimized independently from all others.

For the computation of $E(A)$ and $E'(A)$ we note that all involved terms may be precomputed as soon as the partition is available, except for the estimates $\chi_{A,i}$, which require iterative decoding of f_{T_k} at each stage k. For this repeated decoding fast update methods can be used [5].

5. RESULTS AND CONCLUSION

We computed optimal progressive fractal codes w.r.t. collage errors E and E' for the Lenna image (512×512) originally compressed to 0.5 bits/pixels yielding a compressed file size of 16 KB. The target rate constraints were defined as $m_k = 16k/100$ KB, $k = 1, \ldots, 100$. The results are shown in Figure 1 which displays the achieved image reconstruction quality as the percentage of the received code increases. The two top curves display the ideal case of optimal progressive coding w.r.t. collage error E. The solid curve shows the collage errors and the broken curve slightly below it is for the corresponding reconstruction error. This is only an ideal result, since the decoder cannot attribute the received code to the parameters without additional side information that is not included in the bit rate. Therefore, we may regard the top curves as upper limits of the achievable performance in progressive coding. A practical result is optimal progressive coding w.r.t. collage error estimates E' and shown by the third curve. It is about 2 to 3 dB PSNR below the upper bound of the ideal progressive coder. The last curve at the bottom in Figure 1 shows the poor performance when using the normally ordered bit-stream, where the bits for the

$n = 8.6\%$, PSNR=20.22 dB

$n = 24\%$, PSNR=25.97 dB

$n = 54\%$, PSNR=30.15 dB

$n = 100\%$, PSNR=33.52 dB

Fig. 2. Four sample decodings from the optimal progressive fractal code. n is the size of the partial codes. The PSNR is for the reconstruction error.

parameter tuples (D_i, s_i, o_i), $i = 1, \ldots, n_R$ are placed one after the other in a fixed order determined by the partition. The reconstructed images at four stages of the practical progressive code are shown in Figure 2.

Our work motivates an improved bit allocation of fractal codes simply by letting an optimal progressive coder truncate the output code at a certain percentage of the total rate. Figure 3 shows the results for a range of compression ratios for the test image and parameter settings as above. The coder outputting only 80% of the original rate performed best, yielding gains in PSNR up to 0.6 dB.

Let us summarize. We designed an optimal progressive fractal image coder in the spatial domain. A Lagrange optimization based on performance estimates determines an optimal order of the code bits. The decoder recovers this order and produces high quality reconstructions early on. Although we used quadtree partitions, the method applies to other partitions as well.

Acknowledgment. We thank Michael Hiller for programming work. Our research is supported by grant Sa449/8-1 of the Deutsche Forschungsgemeinschaft (DFG).

6. REFERENCES

[1] Barthel, K. U., Voyé, T., Noll, P., *Improved fractal image coding,* in: Proc. PCS'93, Lausanne, March 1993.

[2] Caso, G., Kuo, C.-C. J., *New results for fractal/wavelet image compression,* in: Proc. SPIE VCIP'96, Vol. 2727, pp. 536–547, 1996.

[3] Davis, G. M., *A wavelet-based analysis of fractal image compression,* IEEE Trans. Image Processing 7 (1998) 141–154.

[4] Fisher, Y., *Fractal Image Compression — Theory and Application,* Springer-Verlag, New York, 1994.

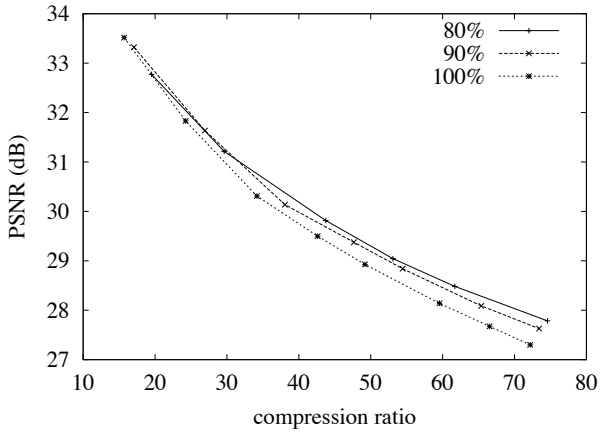

Fig. 3. Improved rate-distortion performance by optimal progressive coders that truncate fractal codes at a given percentage of the total rate.

[5] Hamzaoui, R., Saupe, D., Hiller, M., *Fast code enhancement with local search for fractal image compression,* in Proc. IEEE ICIP-2000, Vancouver, Sept. 2000.

[6] Li, J., Lei, S., *An embedded still image coder with rate-distortion optimization* , IEEE Trans. on Image Processing 8 (1999) 913–924.

[7] Lu, N., *Fractal Imaging,* Academic Press, 1997.

[8] Shapiro, J. M., *Embedded Image Codign Using Zerotrees of Wavelet Coefficients*, IEEE Trans. on Signal Proc. 41 (1993) 3445–3461.

[9] Øien, G. E., Lepsøy, S., *A class of fractal image coders with fast decoder convergence,* in [4].