A PATTERN MATCHING APPROACH TO IMAGE COMPRESSION

Mikhail J. Atallah and Wojciech Szpankowski

Dept. of Computer Science
Purdue University
W. Lafayette, IN 47907
U.S.A.

Yann Génin

Ecole de la Météorologie
42, av. Coriolis
31057 Toulouse
FRANCE

ABSTRACT

We propose an image compression scheme based on approximate pattern matching, that we name *Pattern Matching Image Compression* (PMIC). We give new, efficient algorithms for performing computations motivated by this scheme, and describe the compression ratios experimentally obtained. The main idea is a lossy extension of the Lempel-Ziv data compression scheme in which one searches for the longest prefix of an uncompressed image that *approximately* occurs in the already processed image. It is enhanced with several new features such as searching for reverse approximate matching, recognizing substrings in images that are additively shifted versions of each other, introducing a variable and adaptive maximum distortion level, and so forth. Our scheme is competitive with JPEG and wavelet compression for graphical and photographical images, and it is provably suboptimal under some probabilistic assumptions concerning an image.

1. INTRODUCTION

Data compression based on *exact* pattern matching can be traced back to seminal papers of Lempel and Ziv [9, 10], but recently there has been a resurgence of interest in this type of data compression. It was known for a long time that pattern matching based data compression is very attractive for text compression. A natural question that arises is whether lossy extensions of the Lempel-Ziv scheme are asymptotically optimal, and whether they might be of practical interest. In this paper, we believe we can give an affirmative answer to the latter question (cf. also [2]).

Łuczak and Szpankowski [5, 6] (cf. also [7]) recently constructed a simple, computationally attractive lossy data compression based on approximate pattern matching. The main idea was to search for the longest prefix of the uncompressed file that approximately occurs in the already compressed file (the so called "training sequence" or "database sequence"). They established that under some probabilistic assumption concerning a sequence, the compression ratio is asymptotically equal to the so called *generalized Rènyi entropy* (which is smaller than the optimal rate-distortion function, but not by much).

Research supported by the National Science Foundation under Grant CCR-9202807, and NSF Grants NCR-9415491 and CCR-9201078, and in part by NATO Collaborative Grant CGR.950060.

It must be stressed that the scheme we shall propose in this paper, henceforth called *Pattern Matching Image Compression* (PMIC for short), is a substantial modification of the basic idea described above and analyzed in [6]. A straightforward implementation of the basic scheme on real images (structured data) seems not to be attractive from a practical point of view, so that the enhancements we describe in what follows play an important role in the quality of the experimental results we obtained. These include searching for reverse approximate matching, recognizing substrings in images that are additively shifted versions of each other, making the maximum distortion level variable and adaptive, and so forth. We also give asymptotically efficient algorithms for performing computations arising in an implementation of the above scheme.

Our general practical conclusion can be summarized as follows: The proposed scheme achieves compression ratios comparable to JPEG, wavelet compression (implementation based on [3]), and better than fractal image compression (implemented according to [4]). PMIC works particularly well for images with high frequencies (e.g., containing sharp edges, etc). The compression time is slower than transform based methods such as JPEG and wavelet (at least in the current implementation), but decompression time seems to be the fastest possible due to the fact that our decompression consists mostly of reading and writing data without any processing (except that occasionally it performs one addition operation between the read and the write).

2. THEORETICAL RESULTS

We review and slightly generalize some results of Łuczak and Szpankowski [5]. Consider a stationary and ergodic sequence $\{X_k\}_{k=1}^{\infty}$ taking values in a finite alphabet $\mathcal{A}$ (think of $\{X_k\}$ as an image scanned row by row). For image compression the alphabet $\mathcal{A}$ has size $|\mathcal{A}| = 256$. We write X_m^n to denote $X_m X_{m+1} \ldots X_n$, and for simplicity $X^n = X_1 \ldots X_n$. We also use $P(X^n)$ for the probability of the n-tuple X_1^n. We encode X_1^n into a compression code, and the decoder produces an estimate $\hat{X}_1^n$ of X_1^n. We assume that the reproduction alphabet $\hat{\mathcal{A}} = \mathcal{A}$. As a fidelity measure, we consider any distortion function $d(\cdot,\cdot)$ that is subadditive and that is a single-letter fidelity measure in the sense that $d(x^n, \hat{x}^n) = \frac{1}{n}\sum_{i=1}^{n} d(x_i, \hat{x}_i)$. Examples of fidelity measures satisfying the above conditions are: *Hamming distance*, where $d(x_i, \hat{x}_i)$ equals one if $x_i = \hat{x}_i$ and equals zero otherwise, and the *square error distortion* where $d(x_i, \hat{x}_i) = (x_i - \hat{x}_i)^2$.

As in Łuczak and Szpankowski [5, 6] we define the depth L_n as the length of the longest prefix of X_{n+1}^{∞} that approximately occurs in the database. More precisely, for a given maximum distortion level $0 \le D \le 1$ we define: *Let L_n be the length k of the longest prefix of X_{n+1}^{∞} for which there exists $1 \le i \le n-k+1$ such that $d(X_i^{i-1+k}, X_{n+1}^{n+k}) \le D$.*

A variable-length compression code can be designed based on L_n. The code is a pair (`pointer to` i, L_n). We need $\log n + \log L_n$ bits for the above code. The compression ratio r can be approximated by

$$r = \frac{\text{overhead information}}{\text{length of repeated subword}} = \frac{\log n + \log L_n}{L_n} \tag{1}$$

and to assess quantitatively r we need to know the behavior of L_n.

The probabilistic behavior of L_n depends on a generalized entropy. We define a D-ball $B_D(w_k)$ with center $w_k \in \mathcal{A}^k$, as $B_D(w_k) = \{x_1^k : \ d(w_k, x_1^k) \le D\}$, and we write $P(B_D(X_1^k))$ for the probability of such a ball.
Definition. (Generalized Renyi Entropy) For a fixed $D > 0$ let

$$r_0(D) = \lim_{k\to\infty} \frac{-E \log P(B_D(X_1^k))}{k} \ , \tag{2}$$

provided the above limit exist. □

Łuczak and Szpankowski [6] proved that the above limit exists in the so called mixing probabilistic model defined as: *Let $\mathcal{F}_m^n$ be a σ-field generated by $\{X_k\}_{k=m}^{n}$ for $m \le n$. There exists a function $\alpha(\cdot)$ of g such that: (i) $\lim_{g\to\infty} \alpha(g) = 0$, (ii) $\alpha(1) < 1$, and (iii) for any m, and two events $A \in \mathcal{F}_{-\infty}^{m}$ and $B \in \mathcal{F}_{m+g}^{\infty}$ the following holds* $(1-\alpha(g))P(A)P(B) \le P(AB) \le (1+\alpha(g))P(A)P(B)$. Then, the following result can be established (cf. [6]):

Theorem. *For a subadditive distortion measure, the following holds*

$$\lim_{n\to\infty} \frac{L_n}{\log n} = \frac{1}{r_0(D)} \qquad (\mathrm{pr.}) \tag{3}$$

provided the mixing coefficient $\alpha(g)$ satisfies $\alpha(g) \to 0$ as $g \to \infty$, and the rate of convergence of $\log P(B_D(X_1^n))/n$ *is at least as good as $O(1/n^{1+\delta})$ for some $\delta > 0$. Furthermore, the compression ratio as defined in (1) becomes*

$$r = r_0(D) \quad (\mathrm{pr.}) \ . \tag{4}$$

In general $r_0(D) \ge R(D)$ where $R(D)$ is the optimal rate distortion, however, for small and moderate D they do not differ too much (cf. [5, 6]).

3. ALGORITHMIC RESULTS

The main algorithmic challenge is how to compress efficiently an image. (The corresponding decompression algorithm mostly *copies* and *reads*, and thus is trivially very fast.) We henceforth assume that the image consists of $N \times N$ pixels, and we use n to denote the length of the "database sequence" (in our implementations n will be proportional either to N or to $\log N$).

Let x_1^n denote the database, $y_1^m = x_{n+1}^{n+m}$ denote the yet uncompressed file, and suppose the alphabet is $\mathcal{A} = \{1, 2, \cdots, V\}$. The main algorithmic problem is that of finding an efficient way of computing the longest prefix of y_1^m that approximately occurs (modulo an addition shift by a constant) in x_1^n. A crucial building block we need for this is an algorithm which tests whether one specific prefix y_1^k of y_1^m almost-occurs in x_1^n, i.e., whether there is a position i in x_1^n for which $d(x_i^{i+k-1}, y_1^k) = \frac{1}{k}\sum_{j=1}^{k} d(x_{i+j-1}, y_j - \bar{\delta}) \ \le \ D$, where $\bar{\delta} = \frac{1}{k}\sum_{j=1}^{k}(x_{i+j-1} - y_j)$. Data compression is possible by avoiding storing y_1^k explicitly, and instead storing the additive shift $\bar{\delta}$ together with a (*pointer*, *length*) pair $= (i, k)$ (so that, at decompression, one would approximate each y_j by $x_{i+j-1} + \bar{\delta}$, $1 \le j \le k$). It clearly suffices to compute a *score vector* $C(1), \cdots, C(n-k)$ where $C(i) = \sum_{j=1}^{k} d(x_{i+j-1}, y_j - \bar{\delta})$.

For the square error distortion, we have an algorithm for computing the $C(i)$'s in time $O(n \log k)$. The idea is to expand the equation defining the $C(i)$'s into 10 terms, and to show that the time complexity of computing these 10 terms is dominated by convolution computations that take $O(n \log k)$ time (cf. [1] for details).

For the Hamming distance, we follow a different approach from the computation of the $C(i)$'s. Instead, for given k and i, the almost occurrence of $y_1 \cdots y_k$ at position i in x^n is determined by computing the modulus of the following function

$$F_k(i) = \sum_{j=1}^{k} e^{2\pi V^{-1}\sqrt{-1}(x_{i-1+j}-y_j)} .$$

We can show that $y_1 \cdots y_k$ almost occurs in x^n if the modulus $|F_k(i)|$ is close to k. Therefore, for a given k, determining whether $y_1 \cdots y_k$ almost occurs in x^n can be done by computing the $F_k(\cdot)$ vector and checking whether the modulus of $F_k(i)$ is within a factor α of k *for some* i, where the parameter α is close to 1 (say, $\alpha = 0.95$). We compute the $F_k(\cdot)$ vector in time $O(n \log k)$ (cf. [1] for details).

In either case (square error or Hamming), the above building block is used a logarithmic number of times in a forward binary search for the largest k for which the test returns YES when given x_1^n and y_1^k as inputs. It is strictly speaking inaccurate to use binary search, because the answer might be NO for k and yet that does not preclude that it might be YES for a bigger value than k. This fortunately does not occur often, and can easily be accounted for in the implementation by checking a few additional matchings in the modified binary search. This approach turns out to be a reasonable approximation.

The algorithms currently implemented, include two versions of an essentially brute force approach. One version uses as database the previous f rows (where f is a small integer, no more than 8) and results in an $O(N^3)$ time overall compression time. The second version cuts down the compression time by observing that: (i) Since Theorem shows that the length L_n of the prefix we seek is, with high probability, $O(\log N)$, it is reasonable to restrict the search to a prefix of length $O(\log N)$; and (ii) Since most similarities occur within close proximity, only a fixed number of positions in the f previous rows need to be checked for the almost-occurrence of the length-L_n prefix. This reduces the total worst-case complexity is $O(N^2 \log N)$ and the average case complexity is $O(N^2)$. This "shortening" of the database and usage of "locality" lead to a deterioration of the compression ratio, but our experiments indicate that for most images the deterioration is slight. Implementing the non-brute-force approaches described earlier will decrease the compression times but should not have much impact on the compressions ratios achieved.

Figure 1: Comparison of "San Francisco" image compression: (top) JPEG with the average gray-level error (AGLE) of 4.6 and compression ratio of 8.5 ($\sim$ 1 bpp); (middle) Wavelet based compression with AGLE=3.56 and compression 6.82 (1.2 bpp); (bottom) PMIC with AGLE=6.4 and compression ratio 8.3 ($\sim$ 1 bpp).

4. EXPERIMENTAL RESULTS

Our current implementation uses the quadratic distortion measure. It differs from the basic idea in that it contains the following enhancements that are crucial for achieving good compression ratio:

Additive Shift. It is reasonable to expect that in an image there are several substrings, say x^m, y^m, that differ only by an almost-constant, that is, $y_i = x_i + c$ for $1 \leq i \leq m$ where c is almost-constant in that its variation with i is small. In such a case we store the average difference c as well as a $(pointer, length)$ pair, which enables us to recover y_i from x_i (instead of storing y_i explicitly).

Reverse Prefix. We check for similarities between a substring in the uncompressed file and a *reversed* (i.e., read backward) substring in the database.

Variable Maximum Distortion. It is well known that human vision can easily distinguish an "undesired" pattern in a low frequency (constant) background while the same pattern might be almost invisible to humans in a high frequency (quickly varying) background. Therefore, we used a low value of maximum distortion, say D_1, for a slowly varying background, and a high value, say D_2, for a quickly varying background (we automatically recognize these two different situations by using the derivative of the image).

We also used other, standard techniques such as run-length coding. In order to assess the quality of our PMIC scheme, we performed a series of experiments, and compared PMIC with the standard JPEG (UNIX implementation), wavelet compression (cf. [3]), and also fractal compression (implemented as described in [4]). Since PMIC compression was systematically superior to fractal, we concentrate here on comparing PMIC with JPEG and wavelet compression. In Figure 1 the "San Francisco" image is shown. It has two features of interest to us: First, the sky has almost constant (low frequency) background. Secondly, buildings have sharp edges (high frequency). We apply an adaptive level of distortion (i.e., variable-D) in order to obtain a high quality picture. We also observed that PMIC gives the best quality for sharp edges (cf. [1] for more extensive studies). It turns out that PMIC works particularly well for "graphic" images (e.g., cartoons). In Figure 2 we compare JPEG, the wavelet compression approach and PMIC for such an image.

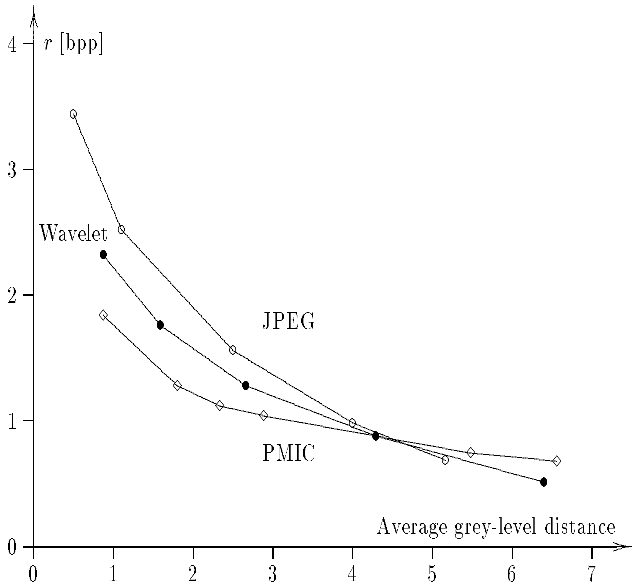

Figure 2: Compression ratio r [bpp] versus the average greay-level error (AGLE) for JPEG (∘), wavelet transform (•), and PMIC (◇) of a cartoon image.

5. REFERENCES

[1] M. Atallah, Y. Genin, and W. Szpankowski, Pattern Matching Image Compression: Algorithmic and Empirical Results, Purdue University, CSD-TR-95-083, 1995.

[2] C. Constantinescu and J. Storer, Improved Techniques for Single-Pass Adaptive Vector Quantization, *Proc. of the IEEE*, 82, 933-939 (1994).

[3] R. DeVore, B. Jawerth, and B. Lucier, Image Compression Through Wavelet Transform Coding, *IEEE Trans. Information Theory*, 38, 719-746 (1992).

[4] Y. Fisher, E. Jacobs, and R. Boss, Fractal Image Compression Using Iterated Transforms, in *Image and Text Compression* (ed. J. Storer), 35-62, Kluwer Academic Publishers, Boston (1992).

[5] T. Łuczak and W. Szpankowski, A Lossy Data Compression Based on String Matching: Preliminary Analysis and Suboptimal Algorithms, *Proc. Combinatorial Pattern Matching*, Asilomar, LNCS 807, 102-112, Springer-Verlag (1994).

[6] T. Łuczak and W. Szpankowski, A Suboptimal Lossy Data Compression Based in Approximate Pattern Matching, Purdue University, CSD-TR-94-072 (1994); also *IEEE International Symposium on Information Theory*, Whistler, (1995).

[7] Y. Steinberg and M. Gutman, An Algorithm for Source Coding Subject to a Fidelity Criterion, Based on String Matching, *IEEE Trans. Information Theory*, 39, 877-886 (1993).

[8] A. Wyner and J. Ziv, Some Asymptotic Properties of the Entropy of a Stationary Ergodic Data Source with Applications to Data Compression, *IEEE Trans. Information Theory*, 35, 1250-1258 (1989).

[9] J. Ziv and A. Lempel, A Universal Algorithm for Sequential Data Compression, *IEEE Trans. Information Theory*, 23, 3, 337-343 (1977).

[10] J. Ziv and A. Lempel, Compression of Individual Sequences via Variable-rate Coding, *IEEE Trans. Information Theory*, 24, 530-536, 1978.