A FAST ALGORITHM FOR ENTROPY ESTIMATION OF GREY-LEVEL IMAGES[†]

Salvatore D. Morgera Jihad M. Hallik

Department Of Electrical Engineering, McGill University, Montréal, Québec

Abstract

This paper examines an efficient approach to the calculation of the entropy of long binary and nonbinary one-dimensional information sequences. The entropy calculation is accomplished in time linear in the sequence length. The method is expanded to estimate the entropy of grey-level images which, under raster scanning, may be represented as one-dimensional information sequences. The entropy estimate obtained depends on the image scanning method employed, and, consequently, to achieve greater reduction in bit rate, the scanning should be done in the direction of highest adjacent pixel statistical dependence. Depending on the image statistics, it is shown that uniform luminance requantization of an image may not lead to an appreciable reduction in bit rate. The algorithm discussed can be applied to areas such as image compression and string entropy estimation in genetics.

1 INTRODUCTION

Entropy is one of the most fundamental and revealing quantities that can be associated with a stochastic information sequence. An accurate estimate of the entropy provides an indication of the amount of redundancy contained in the sequence and, consequently, an upper bound on the data compression possible. This statistical redundancy, which is related to the correlation and predictability of the data, can be removed without destroying any information. In this work, we examine an efficient approach to the estimation of the entropy of an information sequence and extend the method of entropy estimation of binary sequences introduced by Fahner and Grassberger [2] to the estimation of the entropy of grey-level images.

Suppose that we have an $(M \times N)$-dimensional monochrome image X, where each pixel can take one of 2^k luminance values with k equal to the number of bits/pixel. Let a luminance value x have probability of occurence $P(x)$ in the image and be assigned a codeword length of $L(x)$ bits. The average codeword length for the image is given by

$$\overline{L} = \sum_x L(x)P(x) \quad \text{bits/pixel.} \tag{1}$$

The entropy, $H(X)$, is defined as

$$H(X) = -\sum_x P(x)\log_2 P(x) \quad \text{bits/pixel} \tag{2}$$

and provides a lower bound on $\overline{L}$, i.e., $H(X) \leq \overline{L}$ for all decodable, variable length codes that code pixels *independently* of one another [4]. Further bit-rate reduction is possible if pixels are coded and transmitted in *blocks* of n samples, instead of one at a time. This reduction is achieved by taking any correlation that may exist between the n pixels in the block into account. In this case, it is appropriate to express the lower bound on $\overline{L}$ in terms of the block entropy, H_n, which depends on the block size and is given by

$$H_n(\mathbf{X}_n) = -\sum_{\mathbf{X}_n} P(\mathbf{X}_n)\log_2 P(\mathbf{X}_n) \quad \text{bits/block,} \tag{3}$$

where $\mathbf{X}_n = (x_1 \ldots x_n)$ and the x_i's are k bit quantized values. It is not difficult to show that

$$\frac{1}{n}H_n(\mathbf{X}_n) \leq H(X) \quad \text{bits/pixel.} \tag{4}$$

Now, a lower bit-rate is possible without any loss of information using *conditional coding* which is lower bounded by $H(x_n|x_1,..,x_{n-1})$ [4], where we have

$$H(x_n|x_1,..,x_{n-1}) \leq \frac{1}{n}H_n(\mathbf{X}_n) \leq H(X) \quad \text{bits/pixel.} \tag{5}$$

Taking the limit as n becomes large of the conditional entropy, we define h, as in [2], as

$$h = \lim_{n\to\infty} h_n, \quad h_n = H_{n+1} - H_n \quad \text{bits/pixel.} \tag{6}$$

[†]This research was supported by grants from the Canada Natural Sciences and Engineering Research Council (NSERC) and the Québec Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR). The authors are with the Information Networks and Systems Laboratory, Department of Electrical Engineering, McGill University, McConnell Engineering Building, 3480 University Street, Montréal, Québec, Canada H3A 2A7.

0-8186-6715-X/94 $4.00

The derivation of h_n as the conditional entropy is given in the Appendix. The proof that the conditional entropy is less than or equal to the block entropy can easily be verified from the results given in [1].

Since the computational time for the calculation of H_n in (3) grows exponentially with n, this approach to estimating the conditional entropy from block entropies is not suitable for large block lengths. A considerably faster algorithm is required and is discussed in the next section.

2 ENTROPY ESTIMATION OF SEQUENCES

Fahner and Grassberger introduced a method for measuring the entropy of binary sequences using a suffix-tree construction algorithm (*Algorithm M*), previously developed by McCreight [3]. Given the one-sided infinite string $S = (s_1\, s_2\, ...)$, define its one-sided infinite substrings as $S_i = (s_i\, s_{i+1}\, ...)$, where $i = 1, 2, ..., N$, and its finite substrings as $S_{i,k} = (s_i\, ...\, s_{i+k-1})$. The algorithm finds, for each i, the largest k such that $S_{i,k} = S_{j,k}$, for all j between 1 and N. The entropy is then estimated as

$$h = \lim_{n\to\infty} h_n, \quad h_n = \frac{\log_2 N}{\overline{k}_{max}}, \tag{7}$$

where the average maximal k, $\overline{k}_{max}$, is defined as

$$\overline{k}_{max} = \frac{1}{N}\sum_{i=1}^{N} k_i, \quad k_i = \max_j\{k | S_{i,k} = S_{j,k}\}. \tag{8}$$

A straightforward approach to the computation of $\overline{k}_{max}$ in (8) would simply be to attempt to match a substring against the main string for each i. This is a very slow approach, especially as N becomes large, since the computational time does not grow linearly with the sequence length. In order to efficiently calculate the average maximal k's, *Algorithm M* is used to build an "auxiliary index" to facilitate the searches for string repetitions. This index structure can be constructed in time linear in the length of the main string and enables substring searches to also be carried out in time linear in the length of the substring.

2.1 Algorithm M

The algorithm builds a tree structure T (auxiliary index) of the information sequence S, where each path from the root to a terminal node in T uniquely specifies a suffix of S. The last element of S may not appear elsewhere in S, so as to guarantee the existence of a unique terminal node for each suffix of S. If S does not satisfy this property, the existing sequence is padded with a new character.

Define $\mathbf{suf}_i$ to be the suffix of S beginning at element position i (where $\mathbf{suf}_1 = \mathrm{S}$). The algorithm starts by building the empty tree T_0 which contains the root node. During step i, a path corresponding to $\mathbf{suf}_i$ is inserted into the tree T_{i-1} to produce the tree T_i. Figure 1 shows the steps involved in building the suffix tree for the sequence *011012*.

Define $\mathbf{head}_i$ as the longest prefix of $\mathbf{suf}_i$ which is also a prefix of $\mathbf{suf}_j$ for any $j < i$. For the sequence *011012*, $\mathbf{head}_4 = 01$. Also define $\mathbf{tail}_i$ as $\mathbf{suf}_i - \mathbf{head}_i$. Now, consider step 4 in Figure 1 which transforms T_3 into T_4. To insert $\mathbf{suf}_4$, which is *012*, the algorithm traces T_3 and discovers that $\mathbf{head}_4$ is equal to *01* and, thus, splits the first child of the root node into two parts, *01* and *1012*, and creates a terminal node corresponding to $\mathbf{tail}_4$, which is equal to *2*. Now suf_1 and suf_4 are described by the two paths from the root node to the two terminal nodes in the subtree to the left of the root node.

To avoid searching for each $\mathbf{head}_i$ completely, and, thus, run in linear time, the algorithm exploits the following Lemma attributed to McCreight.

LEMMA 1. *If* $\mathbf{head}_{i-1}$ *can be written as* $\mathrm{x}\delta$ *for some character* x *and some (possibly empty) string* δ, *then* δ *is a prefix of* $\mathbf{head}_i$.

As a result, suffix links are introduced into the tree structure which point from the node whose path from the root node describes the string $\mathrm{x}\delta$ and point to the node whose path from the root node describes the string δ. These auxiliary links enable the construction of T in time linear in the length of the sequence S. A complete discussion of McCrieght's algorithm can be found in [3]. An application and extension of this algorithm for data compression was introduced by Rodeh *et al* [5].

2.2 Entropy of independent data sequences

When the infromation sequence is made up of independent data symbols, no further data reduction is achieved by coding blocks, as opposed to coding each symbol on its own. Consequently, we expect that the conditional entropy estimate obtained using (7) will be equal to the entropy $H(X)$ of (2), where the latter is easily obtained by building a histogram of the data symbols. Table 1 compares the two entropies of a binary sequence for a different number of points and for two cases: the case of equiprobable ones and zeros, and the case for which the probability of a one is 1/4 and the probability of a zero is 3/4, where the true entropies are 1.0 and 0.8113, respectively.

The results are consistent, although entropy estimation using the suffix tree algorithm and (7) gives slightly lower values. This can be attributed to the observation that, although the binary sequence is considered independent, some correlation might exist since the sequence was generated by a computer algorithm. In addition, the number of points used for the entropy estimate is limited by the memory space available, which, in our case, is about

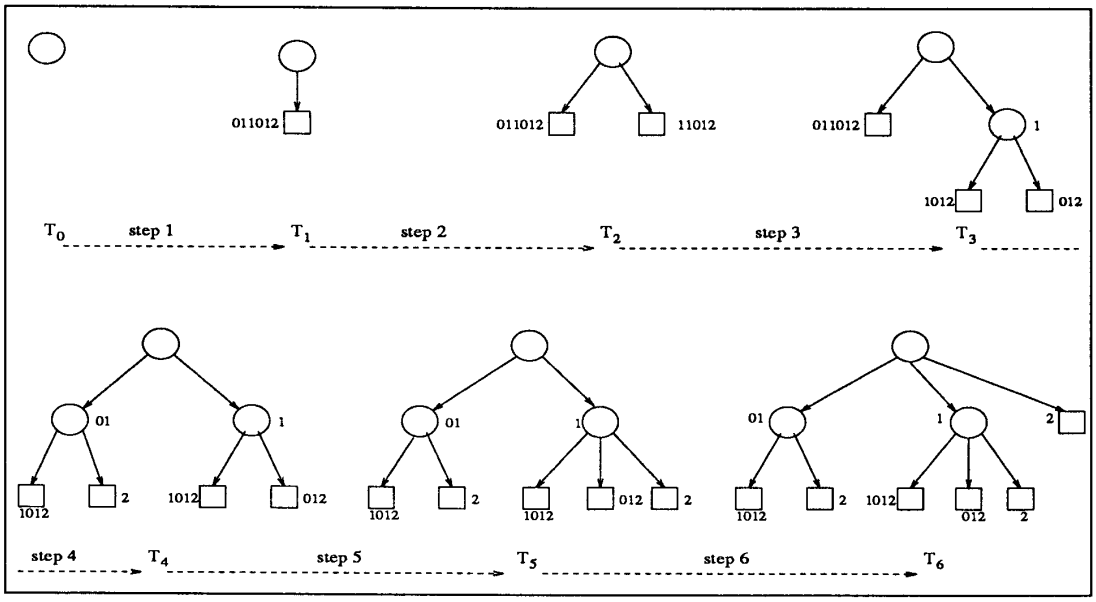

Figure 1: **The suffix tree of the string** *011012*

	equiprobable		$P(1)=\frac{1}{4}$, $P(0)=\frac{3}{4}$	
No. of points	$H(X)$ bits/pixel	h bits/pixel	$H(X)$ bits/pixel	h bits/pixel
1000	0.999896	0.977165	0.851331	0.811807
10000	0.999829	0.967560	0.818952	0.779897
100000	0.999999	0.980705	0.811056	0.778064
400000	0.999996	0.982795	0.811044	0.781222

Table 1: **Entropy of independent binary data**

400000 points (this corresponds to a memory requirement of about 27 MB). Nevertheless, the method provides good estimates and is expected to be quite useful when the data is correlated, such as arises in images.

3 ENTROPY OF IMAGES

The importance of the above algorithm for entropy estimation is its efficiency, considered to be of particular importance when calculating the entropy of images. Since pixel values are usually serially transmitted, the transmitted output is a continuous, one-dimensional information sequence. Consequently, depending on the way the image is scanned, applying the entropy estimation method discussed in the previous section may result in different values for the estimate. To test these variations, two images are used and each scanned horizontally and vertically. The first image is "Lenna" , shown in Figure 2, and the other is "Albert", an image of the face and the upper chest of Albert Einstein, shown in Figure 3.

An estimate of the entropy using (2), which does not take into account the correlation between the pixels, gave 4.182106 bits/pixel and 3.568518 bits/pixel for Figures 2 and 3, respectively. A better estimate of the redundancy present in these images is obtained using (7) and *Algorithm M*. Tables 2 and 3 show the results and computational times of the entropy estimates using *Algorithm M* on a Sparc 10 workstation. The implementation of the algorithm was done in C.

In comparing horizontal and vertical scanning for the *Lenna* image, the entropy estimates agree with the fact that statistical dependence in television pictures is greater in the vertical direction than in the horizontal direction [4]; most televison scanning rasters have a smaller vertical pixel spacing (within a frame) than horizontal pixel spacing, which results in a lower entropy for the sequence

Figure 2: **Lenna; 512x480 pixels (256 grey-levels)**

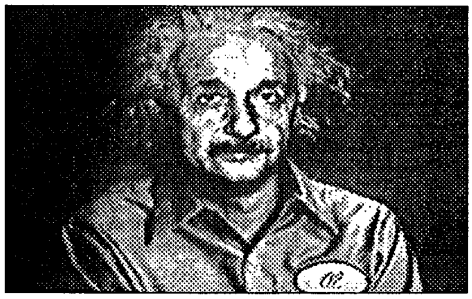

Figure 3: **Albert Einstein; 320x200 pixels (256 grey-levels)**

Scanning Method	Entropy (*bits/pixel*)	Number of pixels	Time (*sec*)
Horizontal scanning	1.589725	245760	16.1
Vertical scanning	1.157746	245760	15.6

Table 2: **Entropy of the "Lenna" image**

Scanning Method	Entropy (*bits/pixel*)	Number of pixels	Time (*sec*)
Horizontal scanning	1.237511	64000	3.1
Vertical scanning	1.323543	64000	3.1

Table 3: **Entropy of the "Albert Einstein" image**

generated by vertical scanning. This property is further exaggerated in this case by the image background which is dominated by vertical edges. On the other hand, horizontal and vertical scanning gave very close entropy estimates for the *Albert* image. Since the number of pixels is relatively small (64000 pixels), the entropy estimation is less accurate. In addition, no edges are present in the background; therefore, we expect the information content of a face, regardless of its orientation, to be approximately the same. This is reflected in the entropy estimates in Table 3.

Another scanning method that we explored was to alternate the scanning direction on consecutive scanned lines. For example, in the case of horizontal scanning, if one line is scanned from left-to-right, the next line is scanned from right-to-left. Intuitively, we expect the entropy of the one-dimensional sequence formed using this scanning method to be lower than the normal raster scan, since a higher statistical dependence should exist between the adjacent pixels in the sequence. This was observed in most of the experiments, although the decrease was slight (0.025 bits/pixel reduction, on average).

3.1 Effects of quantization

Both the *Lenna* and the *Albert* images have 256 uniformly spaced grey-levels in their "natural" format. In this section, the effect on the entropy of requantizing the images in a uniform manner is examined.

In general, it was found that uniformly requantizing the pixel values gave lower entropy estimates. The decrease was not significant for the *Albert* image until 8-level quantization was reached, as can be seen from Table 4. For the *Lenna* image, no significant decrease in the entropy was observed until 16-level quantization was used. This can be explained by examining the grey level image as it is scanned, where we noticed that adjacent levels, in most cases, were either equal or differed by more than 10 grey-levels. This implies that requantization from 256 to 32 levels by uniform quantization has little effect on the entropy. The same applies for the *Albert* image, although, in this case, most of the grey-levels of adjacent pixels were equal or differed by more than 30 levels.

These results indicate that no real benefit, in terms of entropy, is achieved by lowering the number of grey-levels used in images, although individual results may vary. In addition, although it may be impractical to increase the number of grey-levels from 256 to 512 levels, only a slightly higher bit rate is needed, if the results in Table 4 were to be extrapolated.

4 CONCLUSIONS

The algorithm used in this paper for the estimation of the entropy of images is very fast, especially when compared to direct methods that attempt to match substrings to the main strings. The method described takes on the order of seconds, while direct methods take on the order of hours. This large increase in speed, however, is traded off against the additional memory space required which effectively limits the length of the information sequence that can be processed.

Estimating the entropy of images using the method described in this paper is limited in accuracy to the number of points in the sequence, which, for images, is equal to the number of pixels. Consequently, we expect the entropy estimation of small images to be less accurate. In addition, as the number of points used by the algorithm is increased, the entropy estimates converge towards a value, but tend to oscillate about that value. Since the entropy estimate of an image depends on the scanning method used, scanning in the direction of maximum statistical dependence of the pixels will lead to a lower bit rate.

Although the algorithm gives a good estimate of the entropy, and, thus, a good estimate of the redundancy that can be removed, it does not give an indication on how this bound can be achieved. To approach such limits, conditional coding may be used, which, in most cases, gives a lower bit rate when compared to block coding.

Quantization levels	Lenna		Albert Einstein	
	Horiz. Scanning	Vert. Scanning	Horiz.Scanning	Vert. Scanning
256 levels (8-bits)	1.589725	1.157746	1.237511	1.323543
127 levels (7-bits)	1.584278	1.154312	1.237511	1.323543
64 levels (6-bits)	1.580616	1.152054	1.237511	1.323543
32 levels (5-bits)	1.545643	1.131540	1.237511	1.323543
16 levels (4-bits)	1.271702	0.901792	1.237511	1.323543
8 levels (3-bits)	0.941907	0.660785	0.629062	0.564703

Table 4: Entropy of uniformly quantized images

References

[1] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley & Sons, Inc., 1991.

[2] G. Fahner and P. Grassberger. Entropy estimates for dynamical systems. *Complex Systems*, 1:1093–1098, 1987.

[3] E. M. McCreight. A space-economical suffix tree construction algorithm. *J. ACM*, 23(2):262–272, Apr 1976.

[4] A. N. Netravali and B. G. Haskell. *Digital Pictures.* Plenum Press, 1988.

[5] M. Rodeh, V. R. Pratt, and S. Even. Linear algorithm for data compression via string matching. *J. ACM*, 28:16–24, 1981.

APPENDIX

Derivation of the Conditional Entropy

Let $\mathbf{x}_{i,j}, j > i$, denote the sequence $(x_i x_{i+1} \ldots x_j)$. The conditional entropy is given by

$$
\begin{aligned}
h_n &= H_{n+1} - H_n \\
&= -\sum_{\mathbf{x}_{1,n}} p(\mathbf{x}_{1,n+1}) \log_2 p(\mathbf{x}_{1,n+1}) \\
&\quad + \sum_{\mathbf{x}_{1,n}} p(\mathbf{x}_{1,n}) \log_2 p(\mathbf{x}_{1,n}) \\
&= -\sum_{\mathbf{x}_{1,n+1}} p(\mathbf{x}_{1,n+1}) \log_2 p(\mathbf{x}_{1,n+1}) \\
&\quad + \sum_{\mathbf{x}_{1,n+1}} p(\mathbf{x}_{1,n+1} x_{n+1}) \log_2 p(\mathbf{x}_{1,n}) \\
&= -\sum_{\mathbf{x}_{1,n+1}} p(\mathbf{x}_{1,n+1}) \log_2 \frac{p(\mathbf{x}_{1,n+1})}{p(\mathbf{x}_{1,n})} \\
&= -\sum_{\mathbf{x}_{1,n+1}} p(\mathbf{x}_{1,n+1}) \log_2 p(x_{n+1} | \mathbf{x}_{1,n} \\
&= H(x_{n+1} | \mathbf{x}_{1,n})
\end{aligned}
$$

In words, h_n signifies the additional information present in pixel x_{n+1}, given knowledge of all previous pixels, $\mathbf{x}_{1,n} = (x_1 x_2 \ldots x_n)$.