Lossy Image Compression Using Variable Run Length Encoding

Soumya Maitra
Syntel Canada Inc.
E-mail: soumya.maitra@gmail.com

ABSTRACT

Digital image compression is the application of data compression algorithms on digital images in order to reduce redundancy of the image data, thereby enhancing storage and transmission of the concerned image.

Image compression algorithms can be classified under two major categories: lossless and lossy [2][6][7][8]. The former technique preserves image information at the cost of low compression ratios, while the latter achieves high compression ratios at the cost of losing information contained in the image. Lossy algorithms are mostly applied to natural images such as photos in applications where minor loss of fidelity is acceptable to achieve a substantial reduction in bit rate. In this paper the author presents a hybrid approach that extends one of the existing lossless compression schemes with a lossy technique to achieve significant compression ratio while maintaining a high objective fidelity criteria.

Keywords: Lossy Image Compression, Run Length Encoding, Bit Plane Reduction.

1. INTRODUCTION

Run-length encoding is a very simple form of data compression in which runs of data are stored as a single data value and count, rather than as the original run. It is a well-known standard lossless compression technique that relies on repeating sequences of pixel grey levels and is well suited to palette-based iconic images. However, real images (continuous-tone images such as photographs) are mostly composed of slight variations in grey levels instead of contiguous repetitions of identical grey level values. This varying redundancy is a mathematically quantifiable entity. This paper illustrates a novel technique that leverages inter-pixel redundancies with a variable run-length to achieve an acceptable lossy image compression for gray scale images. Our algorithm relies on psycho-visual redundancy in order to apply bit-plane reduction of the uncompressed image to achieve lossy compression. The proposed methodology achieves the above-stated goal while keeping the fidelity criteria at an acceptable level.

The proposed algorithm is designed to exploit known limitations of the human eye, notably the fact that small variations in grey levels are perceived less accurately than small changes in brightness. The proposed compression algorithm is modularized along the following steps of execution, each of which can have scope for future enhancements:

1. Bit-Plane Reduction [1][6]
2. Compute Differential Histogram
3. Apply Pareto Threshold
4. Compute Run-Length & Compress

Compression algorithms require different amounts of processing power to encode and decode. Owing to the upfront computing complexities of the proposed algorithm, compression of a given image is asymmetrical with the decompression scheme, i.e., compression consumes more computational resources than the decompression technique.

2. APPROACH

2.1 Bit Plane Reduction

A bit plane of a digital image is a set of bits having the same position in the respective binary numbers. The first bit-plane gives the roughest but the most critical approximation of grey-level values, and the higher the number of the bit plane, the less is its contribution to the final stage. Thus, adding bit-plane gives a better approximation.

For every sequence of “closely-spaced” grey levels in the un-compressed image, we use two bytes to compress the image. The first byte is used to store the grey level of the run; second byte holds the run. However, the run length encoding is a localized phenomenon, with some regions in the image being eligible for data compression, others not. While decompressing the image, we need to retain this localized information. To distinguish a compressed region in the image from an uncompressed region, we use the least significant bit of each byte.

From [1][6] we know that an image can be reduced into its constituent bit planes (Figure 1), where the least-significant bit plane is a repository of the lowest entropy in the image. In our lossy compression technique using bit-planes we use the bits in the less significant bit-planes as a flag to
distinguish a compressed region in the image from an uncompressed region. Thus, we can afford to lose one bit of information per byte; the least significant bit of each byte used to represent the presence or absence of a run. The least-significant bit holds a grey level of 0 or 1. On a gray scale of 0 to 255 this translates to a loss of 0.39% of information.

Figure 1: Bit plane composition of an image

2.2 Compute Differential Histogram

Next challenge we face is the choice of pixels to compress. Ideally a run-length encoding will compress an identical run of grey-levels. In practice, however, we seldom come across contiguous run of the same pixel value. Instead we get a series of grey-levels that are quite close to each other. These closely-spaced grey-levels can be considered for run-length encoding provided the ‘closeness’ is not diverging. We compute the differential histogram of the image for this purpose.

Computing the differential histogram of the image is basically estimating its differential entropy. Grey levels in a digital image can be considered as one-dimensional. The spacing between one value and the next then gives us a rough idea of the probability density in that region: the closer together the values are, the higher the probability density.

Differential histogram of an image is the plot of differences in the grey levels of adjacent pixels against their frequency of occurrences. Our objective is to identify how “closely-spaced” pixels are within the image. We then apply Pareto’s 80-20 rule on this differential histogram to create a set of grey level differences. The 80% cut-off on the differential histogram gives us the set of grey level differences that defines the solution space in which our compression algorithm executes (Figure 2). A run of contiguous pixels having grey level differences from this set needs to be compressed by the representative mean of the grey levels.

Figure 2: Differential histogram with Pareto cut-off

2.3 Apply Pareto Threshold

The grey-level histogram of an image is an estimated probability density function of the grey-levels of the pixels. It can be seen from the probability density function (Figure 2) above, that the “probability” or fraction of pixels of “closely-spaced” grey levels is rather high, and then decreases steadily as the grey-level differences increase. The Pareto principle (also known as the 80-20 rule) states that, for many events, roughly 80% of the effects come from 20% of the causes. We apply a reverse-Pareto threshold by extending this statement: “80% of pixels in an M x N grey scale image are closely-spaced”, and constitutes the solution space for our algorithm. That is, we consider only those grey-value differences in computing the run-length which fall within the 80% cut-off in Figure 2 above. These grey level differences constitute the set $\Delta$. The 80-20 distribution occurs when the gradient of the line in Figure 2 is $-1$ when plotted on log-log axes of equal scaling.

2.4 Compute Run Length and Compress

Consider an M x N grayscale image that needs to be compressed. Our compression algorithm is going to make two passes over the image. The first pass will compute a differential histogram of the image. The second pass does
the actual compression.

Once we have our set $\Delta$ of grey level differences figured out, we compute the run length of those differences on a row-major scan of the image, starting from [0][0] to [M-1][N-1]. For each pixel and its right-adjacent pixel, if the grey level difference between the two is an element of set $\Delta$, we compute its run length, and the mean of the grey levels. Once the run-length is evaluated, we compress those pixels by two bytes – first one holds the run length, second one holds the mean grey level.

As an illustration consider a run of 8 pixels with the following grey levels with set $\Delta = \{2, 3, 4\}$:

56 58 55 57 59 60 59 56

We will compress it by the following two bytes: 8 and ceil(57.5) where the ceil function returns the next highest integer value by rounding up value if necessary. We can of course use other averaging techniques more relevant to the compression instead of merely taking the arithmetic mean of the run.

The point to be noted here is the applicability of this procedure on the data: A diverging run-length of pixels with valid grey level differences from set $\Delta$ will produce a mean that is inappropriate and breaks the fidelity of the compressed image. This brings up the topic of correlating the information content or the entropy of the run length that needs to be considered for compression. We evaluate the localized entropy based off the differential histogram with the Pareto threshold. This gives us a numerical measure of the uncertainty of the compression procedure. The localized entropy represents a quantitative description of the amount of information in a run-length based on the logarithm of the number of the possible equivalent messages. If the information content of a run can be represented with $N$ possible values (or gray-level differences), and the value $x$ will occur with probability $p(x)$, then the entropy of the image is given by:

$$-\sum_{i=1}^{N} p(x_i) \log(p(x_i))$$

Entropy is typically measured in bits per symbol (grey level). In terms of grey scale digital images, the greater the entropy of the image grey levels, the higher the number of bits required in order to create an adequate representation of the information content.

Ideally, the image to be compressed will have regions that can be compressed, and regions that can’t. This results in localized compression within the image. The computation of the complete histogram of $M \times N$ values takes place in a series of $\log n$ steps $[4][5]$, which is not a bottleneck for our two-pass algorithm.

### 3. ALGORITHM

1. $[M][N] \leftarrow$ Uncompressed grey scale image
2. **Pass I**: Start row-major scan
   a. Compute differential histogram
   b. Apply Pareto threshold on the differential histogram
   c. Evaluate the set of grey level differences
   d. Perform bit-plane reduction
3. **Pass II**: Start row-major scan
   a. Compute variable run length
   b. If variable run length > 3
      i. Compute representative mean of grey levels
      ii. Replace the run by the run length and the mean
      iii. Set the least significant bit to true

### 4. CONCLUSION

The beauty of this algorithm is two-fold. Firstly, it unites a loss-less technique (run length encoding) with a lossy compression approach (bit-plane reduction). Next, it extends the popular run length encoding scheme to varying grey levels in an image. The loss in the resulting image data arises from bit-plane reduction and from replacing run-lengths of varying grey levels with a representative mean of the variation. Since the entropy of the original image is less weighted on the lost bit-plane, our compression algorithm achieves a higher fidelity criterion.

A useful property of our algorithm is that the fidelity criterion can be varied by selecting different averaging techniques. We measure the objective fidelity criteria of our compression algorithm using the root mean square error between the original uncompressed image and the resulting decompressed image.

### 5. FUTURE SCOPE

We have much to explore and achieve in this field.

In order to compress an image we relied on the frequency of occurrence of different grey levels in the image. This called for linearization of the image where the two-dimensional image is first converted into a one-dimensional sequence of grey levels. This paper exploited a row-major scan of the input image to be compressed, where the image is scanned row by row from top to bottom and from left to right within each row. Author suggests studying the correlation between compression ratio and alternative linearization schemes such as column-major scan, diagonal scan, spiral scan, and Peano-Hilbert scan [9].

Fidelity criteria can be both objective and subjective. It is possible to achieve high objective fidelity criteria even with significant loss of information, leading to a low subjective fidelity criteria. The author proposes...
experimenting with statistical techniques on true color images for achieving high-fidelity compression.

Focus could be given towards arriving at a symmetric compression-decompression technique that reduces the computational complexity of the compression algorithm described. It would be appropriate to consider improvising fast algorithms for computing differential algorithms based upon earlier works undertaken in [3][4][5].

Author suggests undertaking a comparative study of applying various averaging techniques and the corresponding compression ratio achieved.

Author has experimented with Pareto Principle. Threshold determination by other statistical means such as Poisson distribution, Point Process Model, etc. would be another area of improvement.

6. REFERENCES