Journal of Kerbala University , Vol. 7 No.3 Scientific . 2009

Two Dimensional Dictionary –Based Lossless Image Compression

Assistant Lecturer Inas R. Shareef
Karbala University
College of Science
Computer Department
Inas_shareef@yahoo.com

الخلاصة

يستخدم ضغط الصور عادة لتقليل المساحة الخزنية كهدف اساسي له. على اية حال، هو مرغوب لتشفير الصورة لقدرته على معالجة نتائج التمثيل مباشرة.

في هذا البحث، يعتمد نظام الضغط المقترح لضغط الصور على طرق الضغط المسندة بالقاموس. حيث تم وصف التصميم والتطبيق لخوارزميات الضغط وفتح الضغط. طرق الضغط المسندة بالقاموس هي من الطرق الشائعة لضغط بيانات الملف. وتعد طريقتي الضغط LZ77، LZ78 ومشتقاتهما هما الاكثر شهرة من هذه الطرق. هذه الطرق تطبق لتقليل الترابط في البيانات احادية البعد، لذلك صممت لضغط النصوص. لذا فهي لا تعطي الفائدة لطبيعة نقاط الصورة، حيث ان النقاط المتجاورة في الصورة هي متقاربة القيم (اي مترابطة). في هذا البحث قدم نموذج ضغط الصورة بدون خسارة المسند بالقاموس ثنائي الابعاد للصورة الرمادية.

تم اختبار نظام الضغط المقترح باستخدام عدة صور ذات احجام مختلفة. حيث تم الحصول على نسب ضغط جيدة من هذا النظام المقترح، معتمداً على طبيعة الصورة. وايضاً تم دراسة المقارنة بين هذه الطريقة المقترحة وطرق اخرى تم ذكرها.

بينت نتائج الاختبار ان اداء الضغط للنظام المقترح افضل من طرق ضغط مبنية على مبدأ القاموس. والنتائج ايضاً بينت تقاربها مع اداء طريقة الضغط لـ JPEG2000 ، عندما تطبق في طور عدم الخسارة، وايضا تمت مقارنتها مع اداء طريقة الضغط لـ JPEG-L ، حيث ان JPEG2000 و JPEG-LS هي طرق الضغط القياسية الحالية.

Abstract

Images compression usually considers the minimization of storage space as its main advantage. It is desirable, however, to code images so that the ability to process the resulting representation directly is possible.

In this paper, a compression system based on Dictionary-Based compression methods is proposed for compressing images. The design and implementation of novel compression and decompression algorithm are described.

Dictionary-Based compression methods are a popular form of data file compression. LZ77, LZ78 and their variants are likely the most famous of these methods. These methods are implemented to reduce the one-dimensional correlation in data, since they are designed to compress text. Therefore, they do not take advantage of the fact that, in images, adjacent pixels are correlated in two dimensions. In this paper, a true two-dimensional dictionary-based lossless image compression scheme for grayscale images is introduced.

The proposed compression method and the compression system are examined using different images with different sizes. Good compression ratios have been obtained from the proposed system, and this depends on image nature. Comparisons between the proposed method and different methods is also given and the proposed algorithm gives better results than the other methods.

Testing results show that the compression performance of the proposed scheme outperforms and surpasses any other existing dictionary-based compression scheme. The results also show that it slightly outperforms JPEG-2000's compression performance, when it operates in its lossless mode, and it is comparable to JPEG-LS's compression performance, where JPEG-2000 and JPEG-LS are the current image compression standards.

1. Introduction

Contemporary computers process and store huge amounts of data. Some parts of these data are excessive. Data compression is a process that reduces the data size by removing the excessive information. Short data sequence is often more suitable, because it reduces the costs.

Compression techniques are classified into two categories [1,2]: (a) Lossless, and (b) Lossy approaches. Lossless techniques are capable to recover the original representation perfectly. Lossy techniques involve algorithms which recover the presentation to be similar to the original one. The lossy techniques provide higher compression ratios, and therefore they are more often applied in image and video compression than lossless techniques. The compression scheme presented in this paper is a lossless scheme.

Among the most popular methods of lossless compression are Dictionary-Based schemes. Dictionary compressors encode a string of data by partitioning the string into many sub-string, and then replacing each sub-string by a codeword. Communication between the compressor and the decompressor is done using messages, each message consists of a codeword and possibly other information. The dictionary in these schemes is the set of every possible codeword. LZ77 [3] and LZ78 [4] are two of the most famous dictionary-based compression schemes. In LZ77, the dictionary is a portion of the most recently encoded data. This is also called the search buffer. Codewords for sub-strings are pointers to the longest match for the sub-string found in the search buffer. Each message consists of the codeword for the sub-string, the length of the match and the code of the next symbol. In LZ78, the dictionary codewords correspond to previously encountered substrings. Each codeword consists of two parts, a pointer to the dictionary and the code of the next symbol.

LZ77, LZ78, and their variants (Lempel-Ziv) methods [5,6], take advantage of the fact that adjacent data values are highly correlated. These dictionary-based schemes are designed to compress text and so only reduce one-dimensional correlations in data. Therefore, they do not take advantage of the fact that, in images, adjacent data values (pixels) are highly correlated in two dimensions.

The dictionary-based scheme presented in this paper is designed to take advantage of the two-dimensional correlation between pixels in grayscale images. It allows for approximate matches since it is designed to compress grayscale images. The aim of this paper, is to propose compression system, that attempts to adapt the one-dimensional Lempel-Ziv compression to suit the two-dimensional nature of images.

The rest of this paper is organized as follows. Section 2 and 3 describes the proposed scheme in details. Section 4 presents the results. Section 5 offers the conclusion of the paper. Finally, section 6 includes the suggestion for future work.

2. The Two-Dimensional Lempel-Ziv Compression

Grayscale Two-Dimensional Lempel-Ziv (denoted as GS-2D-LZ) is an image compression scheme that is based on the popular family of LZ text compression schemes. However, its dictionary is built on the image history in two-dimensions, rather than in a linear fashion. Hence, it can take advantage of the two-dimensional correlations in image data.

We have extended the LZ77 algorithm to two dimensions, thereby taking advantage of the inherent two-dimensional nature of the image. Pixels are still encoded using raster scan order. However, the linear search window, which appears in LZ77, is replaced with a rectangular search region of previously coded pixels.

So in GS-2D-LZ, an image is encoded in raster scan order, one block of pixels at each step. For each block of pixels an *approximate match* is searched for in previously encoded data. The block is encoded as a pointer to the location of this approximate match, along with the dimensions of the match and residual information to ensure that the compression is lossless. If no suitable match can be found, a block of pixels is encoded using the Differential lossless compression scheme. After the entire image has been encoded in this fashion, the match location, match position, residual and difference values are separately encoded using a statistical compression scheme.

2.1. Compression system stages

The proposed system consists of many stages, which are :

1. Transform the input image to grayscale image.
2. Encoding the high and the left boundaries.
3. Search and Encoding.

Figure (1) gives the block diagram of this system, which shows its components.

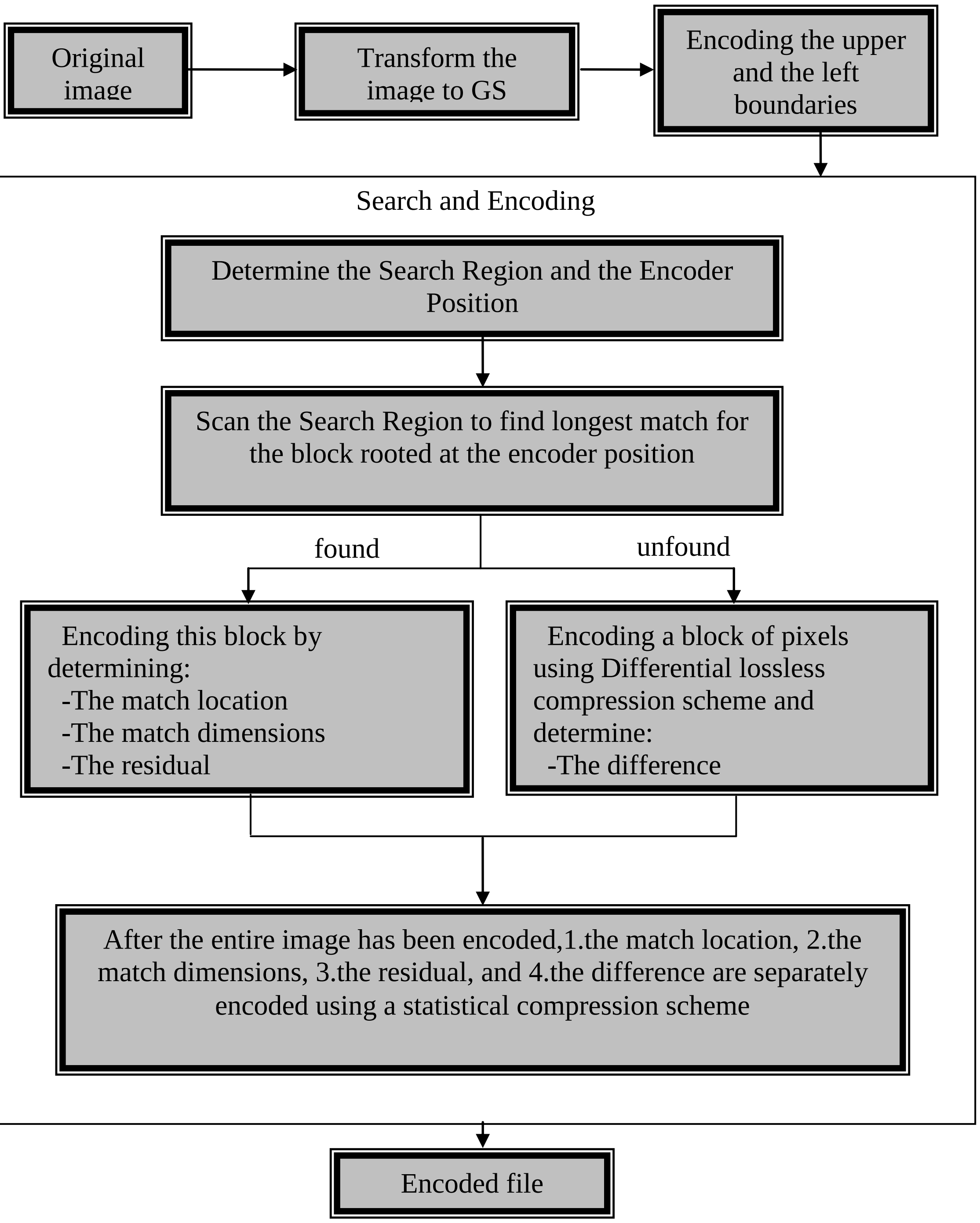
Original image
Transform the image to GS
Encoding the upper and the left boundaries
Search and Encoding
Determine the Search Region and the Encoder Position
Scan the Search Region to find longest match for the block rooted at the encoder position
found
unfound
Encoding this block by determining: -The match location -The match dimensions -The residual
Encoding a block of pixels using Differential lossless compression scheme and determine: -The difference
After the entire image has been encoded,1.the match location, 2.the match dimensions, 3.the residual, and 4.the difference are separately encoded using a statistical compression scheme
Encoded file

Figure (1) The block diagram of the compression system

2.2. The components of GS-2D-LZ compression scheme

The GS-2D-LZ compression scheme includes many components, as the following :

1. Previously coded pixels (region).
2. Search area (search region).
3. Match region.
4. Not yet coded region (block to be coded).
5. Encoder position.

As mentioned, the linear search window, which appears in LZ77, is replaced with a rectangular search region of previously coded pixels.

The search area, in which matches are considered for a block, is rectangular in shape and located above and to the left of the encoder position. The search region is a function of *search-width* and *search-height variables*, which are adjustable parameters that identify the horizontal and vertical search distances, respectively. The search region must be located at previously encoded region.

The match region, located inside the search region, is illustrated in figure (2), a match is a rectangular region, specified with four coordinates: a pair of coordinates, x and y, specify the match position, and another pair of integers, width and height, specify the match.

The block to be coded represents the block not yet coded, rooted at the encoder position and for each block of pixels an approximate match is searched for in previously encoded data (i.e. in search region).

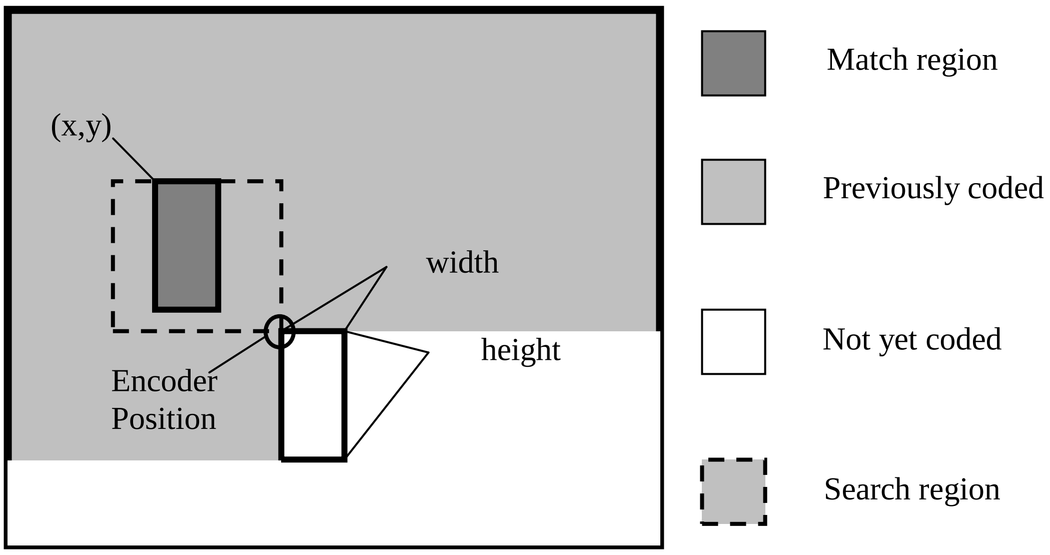

Figure (2) illustrates the components of GS-2D-LZ algorithm

2.3. Search and coding

Before beginning the searching and the encoding, must preparing encoded region, must be prepared to start the search region from it. The upper and the left boundaries are encoded for this target, by using any suitable compression method, and the method used here is Differential Lossless Compression method. The boundaries thickness that are needed for encoding, depends on the search region size. These boundaries will become previously encoded region, as illustrated in figure (3).

When searching for a match of the block rooted at the encoder position, each pixel in the search region represents a block of pixels rooted at the same position. There are (search-width – 1)

× (search-height – 1) unique blocks to be considered (since the block rooted at the encoder position is not a possibility).

For a particular root pixel in the search region, the algorithm calculates the difference between that pixel (in the search region) and the pixel at the encoder position. To be considered as a possible match, the difference of these two pixels can'ot exceed the value of a variable called threshold, which is an adjustable parameter that identifies the maximum allowable error between pixels.

Here it is important to illustrate, this algorithm does not depend on the actual matching or the equivalent match between these two pixels, but depends on the approximate matching. By taking the difference between these two pixels, to make this algorithm efficient and more flexible, and at the same time, do not ignore the difference values between the two pixels, ensure that the compression is lossless.

If a pixel is qualified as a root of a potential match (the difference between these two pixels does not exceed the value of the threshold), the match is then extended to the right as wide as possible using the same criteria, i.e. the match will be extended one pixel to the right, and also the block being encoded will be extended one pixel to the right. Now if the difference between the new two pixels does not exceed the value of the threshold, then the match and the block being encoded will be extended to the right as wide as possible, and so on. in order for the match to be extended one pixel to the right, the difference between corresponding pixels in the potential match and the block being encoded must be less than the threshold.

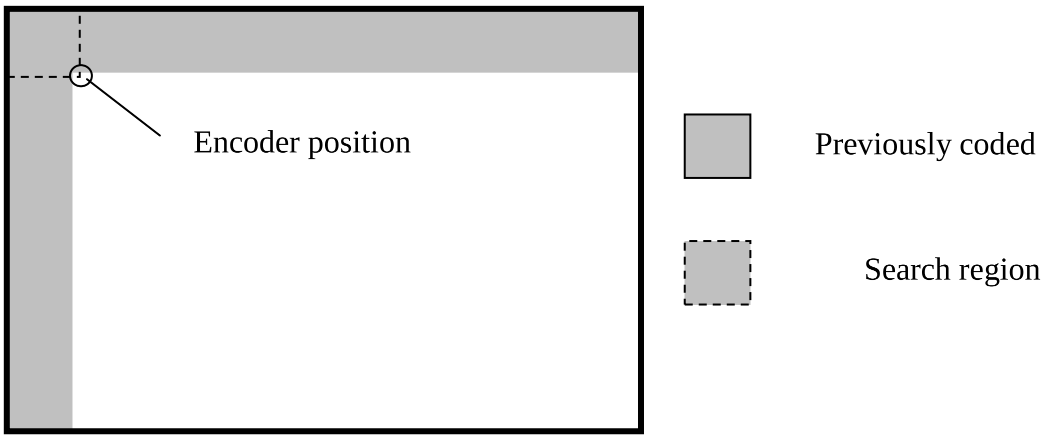

Figure (3) illustrates the relation between the boundaries and the search region

Once a mismatch occurs, the match and the block being encoded are then extended down as far as possible. This step is similar to the case when a match and the block being encoded are extended right.

The match and the block being encoded continue being extended by this manner. When a mismatch occurs in a row extension, the width of the match is simply reduced in order to allow the match to continue extended downwards. Extending the match further downward by this manner will eventually force a reduction of the potential width to be less than 1. When this occurs, the algorithm deems that all possible matches at this root are exhausted and all found matches are evaluated by a condition :

- The match must be large enough that more new pixels than the value of a variable called minimum-match-size are being encoded. The variable minimum-match-size is an adjustable parameter.

Due to the nature of the variable block size scheme that we use, encoded blocks will sometimes overlap with previously encoded pixels. In this situation, it does not matter that these pixels match or not, nor do these pixels count as new pixels.

These steps are repeated by searching each pixel in the search region to find an approximate matches of the block rooted at the encoder position.

After considering every match rooted at each pixel in the search region, the largest match is selected and encoded. In the case where no suitable match can be found, the algorithm encodes a small block of pixels rooted at the position of the encoder, in order to ensure that progress is made. The dimensions of this block are fixed to no-match-block-width × no-match-block-height, where no-match-block-width and no-match-block-height are two adjustable parameters. To keep the algorithm fast and efficient, a differential lossless compression scheme [7] is used to encode these pixels.

2.4. The Adjustable Parameters

The adjustable parameters in the GS-2D-LZ scheme have been empirically chosen, where *search-width, search-height, threshold, minimum-match-size, no-match-block-width, and no-match-block-height are set to 4, 4, 27,18, 5, and 5, respectively.*

The first two parameters (*search-width, search-height)* are used to define the search region size, and the experimental shows that the suitable size is 4×4, because, when extend the search region size this will be due to increase the probability of finding matches, but in the other hand the searching time also increased. In case when shark the search region size, this will be due to decrease the probability of finding approximates.

The threshold parameter value is chosen by experimental and would be equal to 27. In the case when choose lower value this will be increase in the probability of finding approximate matches that have large size, and this will be increase the size of tables which saves the information related to the matches, but when increasing the threshold value, the probability of finding approximate matches will be poor.

The forth parameter (minimum-match-size) value is chosen by experimental and would be equal to 18 (this means the number of pixels in the match block), and must not be lower than 18. When decrease this value, the match block will be not suitable for encoded, and this means that it will be take a space from tables without useful. If this parameter increase, this will be due to inefficient algorithm.

The last two parameters (*no-match-block-width, no-match-block-height)* responsible for giving the size of the mismatch block, and the suitable value is chosen to be (5×5) by experimental, the purpose of this block is to ensure that the progress is made in the algorithm, so when chosen small size this will not due to ensure the main purpose, and when chosen largest size, the work of this algorithm would be poor.

2.5. Data Structures Defined

There are five tables that are used to record the matching information. These tables are called: *Match Flag*, *Match Location*, *Match Dimensions, Residual*, and *Difference.*

The *Match Flag* table contains a Boolean value for each block of pixels, where a value of *true* is recorded in the table when a suitable match for the block is found and *false* otherwise. When a suitable match is found for a block, the position of the match, relative to the block being encoded, is recorded in the *Match Location*. At the same time, the width and height of the block being encoded are recorded in the *Match Dimensions* tables. Moreover, the difference between each pixel of the actual block and the corresponding pixel in the match found for the block is recorded in the *Residual* table. On the other hand, when no suitable match can be found, the *Difference* table is used to hold the difference values that result by compressing a block of pixels by using differential lossless compression scheme.

After the entire image has been scanned, each table is encoded using Huffman compression method.

3. components of the suggested decompression system

Figure (4) illustrates the block diagram for the decompression system.

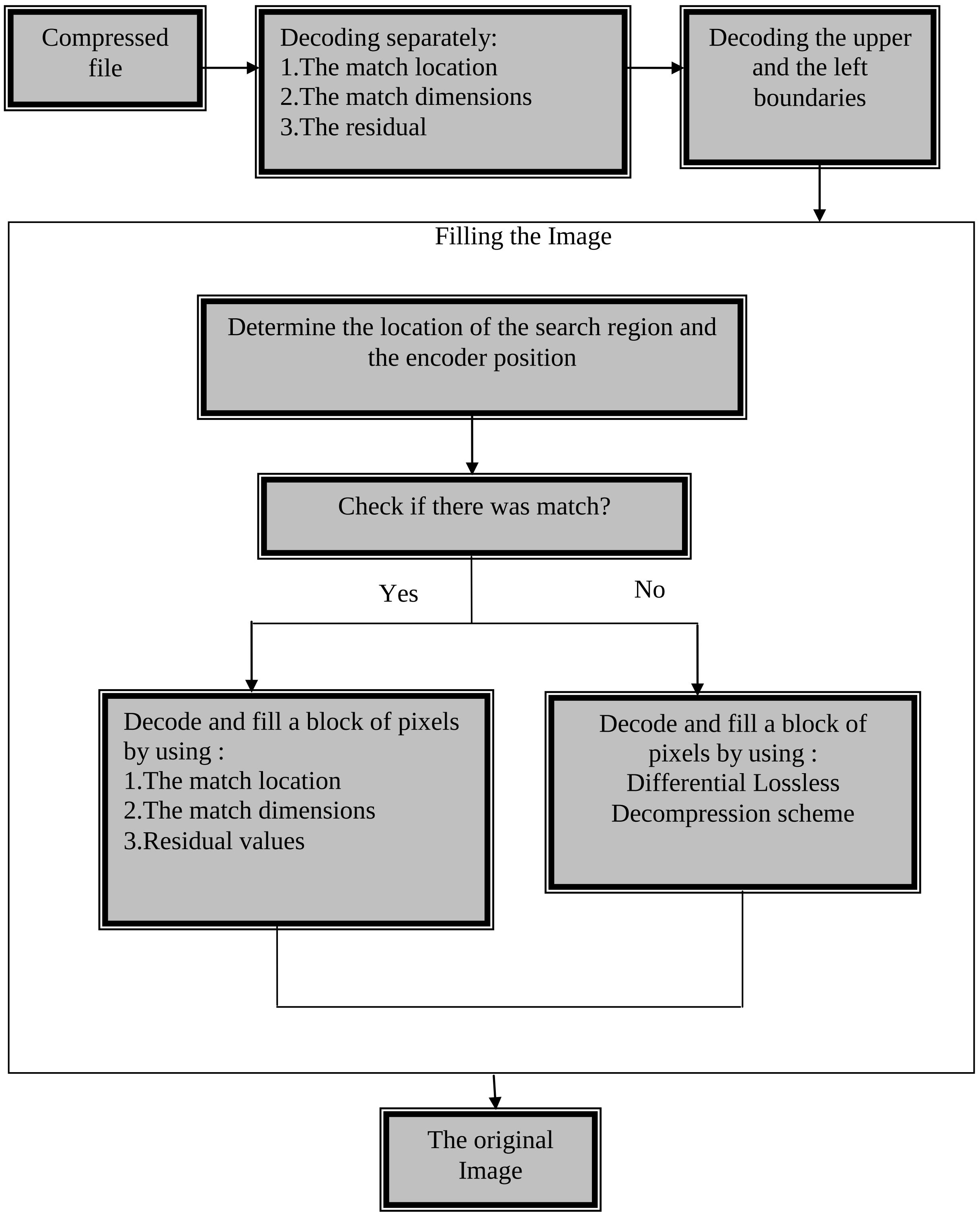
Compressed file
Decoding separately:
1.The match location
2.The match dimensions
3.The residual
Decoding the upper and the left boundaries
Filling the Image
Determine the location of the search region and the encoder position
Check if there was match?
Yes
No
Decode and fill a block of pixels by using :
1.The match location
2.The match dimensions
3.Residual values
Decode and fill a block of pixels by using :
Differential Lossless Decompression scheme
The original Image

Figure (4) illustrates the block diagram for the decompression system

3.1. Two-Dimensional Lempel-Ziv Decompression scheme

The decoding of the GS-2D-LZ is simpler than the encoding. First the five tables : Match Flag, Match location, Match location, Residual, and Difference are decoded with Hoffman decompression scheme.

After these tables have been decoded, the left and the upper boundaries are decoded by applying the differential decompression scheme. Now the filling steps are beginning, and these steps start with determining the search region location and the encoder position (this step is almost like in compression). Hence, the search region starts at the beginning located in the left and high corner of the image, and then moves depending on the next steps of the algorithm. The succeeding step include checking the Match flag table, if the value is true, this means a suitable match for the block was found in the encoding step, so the decoder decodes and fills this block. This process is done by taking the location of the match (that located at the search region) from the Match Location table, and taking the size of the match (and so the block) from the Match Dimension table. Now, reconstructing the block becomes easy by taking the residual values from the Residual table and add to the corresponding pixels at the match block, and filling the block with the results, i.e. each value resulting from the addition represents the corresponding pixel at the block.

IF the value of the match flag table is false, that means no suitable match for the block is found in the encoder step. At this case the decoder decoding a block of pixels rooted at the encoder position, and the dimensions of this block are fixed (at encoding process) to *no-match-block-width* × *no-match-block-height.* And the pixel values of this block can be found by applying the differential lossless decompression scheme using the Difference table.

The steps from determining the search region and the encoder position, are repeated until the entire image is filled by the original values.

Finally the decoder can reconstruct the original image without loss of any information. The decoder does not need to perform any search, and is therefore much simpler in design and implementation than the encoder.

4. Experimental Results

This experiment is concerned with compression ratio , by using a simple ratio formula as in equation (1). The (GS-2D-LZ) scheme described in this paper was tested on a set of 12 different grayscale natural scene images, shown in fig. (3).

$$\text{Compression ratio} = \frac{\text{Uncompressed File Size}}{\text{Compressed File Size}} \qquad (1)$$

For each of these images, the compression performance (in bit per pixel) of the (GS-2D-LZ) scheme was compared to the compression performance of PNG, JPEG2000, AND JPEG-LS schemes, as shown in table (1). Note that, PNG is based on LZ77 scheme, JPEG2000 [26]is the current JPEG lossy compression standard but operated in its lossless mode, and JPEG-LS [29]is the current JPEG lossless compression standard.

The GS-2D-LZ scheme slightly outperforms JPEG2000, when operated in its lossless mode. At the same time, its compression performance is comparable to JPEG-LS's performance. It is worth mentioning that while all other compression schemes (including JPEG2000 and JPEG-LS), are heavily optimized, the proposed scheme is not, since it was implemented just as a proof of concept.

Figure (3): test images

Table(1) Compression performance measured in bits per pixel

Image name	GS-2D-LZ	PNG	JPEG2000	JPEG-LS
Baboon–512x512	3.679	6.01	5.88	5.82
Boats–720x576	4.087	4.33	4.07	3.93
Camera–256x256	4.094	4.67	4.54	4.31
columbia–480x480	4.319	3.92	3.52	3.43
Couple–512x512	3.99	4.88	4.84	4.68
Crowd–512x512	4.175	4.53	4.20	3.91
Lena–512x512	4.148	4.39	4.06	3.99
Man–512x512	3.981	4.93	4.69	4.50
milkdrop–512x512	4.341	3.97	3.77	3.63
peppers–512x512	4.072	4.91	4.63	4.51
Woman2–512x512	4.368	4.98	4.81	4.67
woman1–512x512	3.992	3.77	3.32	3.30
Average	4.103	6.65	4.80	5.18

5. Conclusion

A two-dimensional dictionary based scheme is introduced. In conclusion one may note the following:

1. Experimental results showed that the compression performance of (GS-2D-LZ) scheme outperforms and surpasses any other dictionary based compression scheme. At the same time, it performs slightly better than JPEG2000 and is comparable to JPEG-LS. This implies that dictionary based compression schemes can be as efficient as the current state of the art compression scheme.
2. This work represents new method regarding both concept and application, because, the search process for match block differs from other compression methods in which they divide the image into equal blocks (equal in size block), but here is rectangle region (search region) is used to find the match blocks which would be variables in size.
3. It consists of efficient search process, the control process on the movement of the search region and finding matches are very complex, but at the same time is very flexible and efficient.
4. To improve compression efficiency, we allow the match region to extend beyond the search region, this is because the match region has a variable size, only the upper left hand corner of the match region actually needs to be inside the search region. By allowing these large matches, we can encode large repetitions. This situation appears in two cases:
    a) Horizontally, the match width is extended beyond the search region to overlap with previously encoded region.
    b) Vertically, the match height is often limited, because match regions extending into not yet encoded regions cannot be decoded.

6. Suggestion For Future work

There are a number of directions in which this work could be extended. Some of these directions are:

1. In (GS-2D-LZ) scheme, for each block of pixels (block to be encoded) an approximate match is searched for in previously encoded data. We can use an actual match or an equivalent match instead of an approximate match. In this case we can reduce the number of tables, that's by not saving the Residual table, but at the same time, the probability of finding the match may be reduced.
2. In (GS-2D-LZ) scheme,, we use the differential lossless compression scheme to compress mismatch blocks. We can use the DPCM compression method, which is a member of family of differential encoding compression methods, it is based on the well-known fact that neighboring pixels in an image are correlated. This means that in general an item ai depends on several of its neighbors, not just on the preceding item ai-1 , as in differential method.

References

[1] Fruht B., Smoliar S.W., Zhang H., "Video and Image Processing in Multimedia Systems", Kluwer Academic Publisher, 1996.

[2] Salomon D Deorowicz S., "Universal Lossless Data Compression Algorithm", Ph.D. Dissertation, faculty of Automatic Control, Electronics and Computer since, Silesian University of Technology, April 2003.

[3] Abbas T., "A Hybrid Algorithm for Images Compression", Master thesis, Department of Computer Science, University of Technology, 2004.

[4] Ziv J., Lempel A., "Compression of Individual Sequences via Variable-Rate Coding", IEEE Transactions on Information Theory, Volume 24, No. 5, pp. 530-536, September 1978.

[5] Welch T., "A Technique for High-Performance Data Compression", IEEE Computer, pp 8-19, June 1984.

[6] Zeeh C., "The Lempel Ziv Algorithm", January 2003.

[7] Salomon D., "Data Compression The Complete Reference", Second Edition, Springer 1998.