Proceedings of the 2nd International Conference on Information Technology for Application (ICITA 2004)

How Can Spiral Architecture Improve Image Compression?

Xiangjian He, Huaqing Wang, Tom Hintz, and Qiang Wu

Abstract--**An image file contains a large amount of data. Image compression changes a large image file into a much smaller file. Smaller files require less memory to store, less computer network bandwidth to transfer on the Internet, less time for a computer to process. Image compression becomes important and necessary due to limited computer network bandwidth, computer processor speed and computer storage size. In this paper, we represent an** image on our recently created image structure, called Spiral Architecture. We propose algorithms for image compression based on important features of Spiral Architecture: locality and **uniformity. Locality is a consequence of a special scheme for numbering image pixels which guarantees physical proximity of the pixels with neighbouring addresses. Uniformity refers to uniformly separating image into similar sub-images.**

Index Terms—**Image Compression, Spiral Architecture, image partitioning**

I. INTRODUCTION

NEEDLESS to say, visual information is of vital importance if human beings are to perceive, recognize and understand the surrounding world. With the tremendous progress that has been made in computer power, the corresponding growth in the multimedia market and the advent of the World Wide Web, it is becoming more than ever possible for image to be widely utilized in our daily life. In general, an image file contains much more data than a text file. An image with large amounts of data require much memory to store, takes longer to transfer, and is difficult to process. For example, a black and white image with 256×256 pixels requests about 64 Kilobytes of memory space and more than 6 seconds to download using a Dialup Internet connection. As a consequence, image compression becomes necessary due to the limited communication bandwidth, CPU (for computer Central Processing Unit) speed and storage size. Image compression has been pushed to the forefront in the image processing field.

The research work to be presented in this paper is based on a novel data structure, Spiral Architecture [1], which is inspired from anatomical considerations of the primate's vision [2]. On Spiral Architecture, an image is a collection of hexagonal picture elements [3] as shown in Fig. 1. In the case of human eye, these elements (hexagons) would represent the relative position of the rods and cones on the retina. Each pixel on Spiral Architecture is identified by a designated positive number, called *Spiral Address*. The numbered hexagons form the cluster of size 7^n. The hexagons tile the plane in a recursive modular manner along the spiral direction as shown in Fig. 1. Any hexagonal pixel has only six neighboring pixels which have the same distance to the centre hexagon of the seven-hexagon unit of vision.

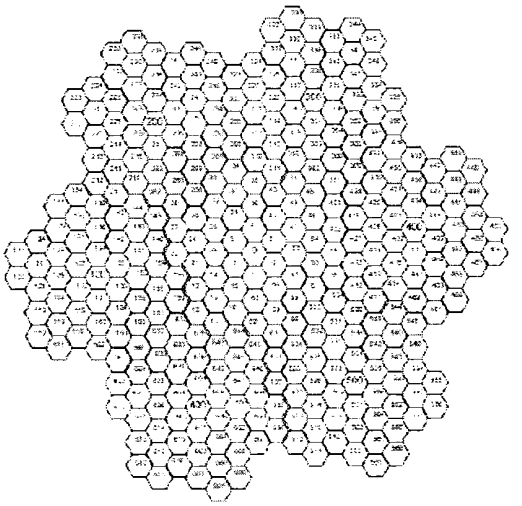


Fig. 1. A collection of 343 hexagonal pixels

In our preliminary research on image compression based on Spiral Architecture [4], we focused on the properties of the hexagonal pixel address labeling scheme. The property of interest was the physical proximity of the hexagonal pixels with neighboring addresses. Rectangular systems may, for instance, have vertical physically adjacent pixels but the address distance is the length of a scan line. It was demonstrated that in the Spiral Architecture, unlike the rectangular system, neighboring pixels, for example, the pixels with addresses 0, 1, 2, 3, 4, 5 and 6, have the similar intensities.

The research reported in our most recently written paper [5] uses the properties of uniform image partitioning, which is a novel image operation developed recently based on Spiral Architecture. On Spiral Architecture, an image can be partitioned into a few sub-images each of which is a scaled down near copy of the original image. Namely, each sub-image holds all the representative intensity information contained in the original. Using such properties, in our work, the points in the original image first are re-allocated into a few groups, sub-images, rather than being unwound in the spiral address

All authors are with the Computer Vision Research Group, University of Technology, Sydney, PO Box 123, Broadway, NSW 2007, Australia (contact telephone: 61-2-95141816, e-mail: {sean.huwang.hintz.wuq}@it.uts.edu.au).
All authors except Huaqing Wang are IEEE members.

ICITA2004 ISBN 0-646-42313-4

order to be another one-dimensional data set as shown in [4]. The similar pixel intensity is found between the corresponding points in the different sub-images. Then, it is possible to choose one sub-image as a reference image and work out the intensity difference between the reference image and other sub-images such that the original image will be stored by recording only the reference sub-image and the intensity difference information thus giving opportunities for better image compression. We have observed that when the partition number is smaller, the shorter bits account for the higher percentage. As the number of partition increase, the longer bits take over the shorter bits to account for the higher percentage. In fact, uniform image partitioning on Spiral Architecture is a unique re-sampling procedure. After such partitioning all the points in the original image are moved to the unique new positions. Thus, the corresponding points in each sub-image, which are on the same positions relative to central point of each sub-image, were close to each other in the original image. But as the number of partition increases, the original distance between these corresponding points in the input image increases such that the difference of the grey scale becomes stronger. Namely, we need more bits to represent such grey scale difference information.

The organization of this paper is as follows. Section II overview the image compression methods. Spiral Architecture is reviewed in Section III. We present image compression schemes based on Spiral Architecture in Section IV. We conclude in Section V.

II. Overview of Image Compression Schemes

There have been a lot of image compression methods developed in the previous few decades. Among them, some has been perfectly matured and some are still under development.

According to whether the compressed file preserves every detail perfectly or allows some loss of data, image compression methods fall into two categories: *lossless compression* and *lossy compression*. We list below the popular image compression techniques which will be adapted in this paper to work with Spiral Architecture (SA).

Lossless methods

The simplest method in this category is called Run-Length Coding (RLC) [6]. RLC had been in use since the earliest days of information theory. It takes advantage of repetitive data. The term run is used to indicate the repetition of a symbol, and the term run-length is used to represent the number of repeated symbols, that is, the number of consecutive symbol of the same value. Instead of encoding the consecutive symbols individually, it is obvious that encoding the run and the run-length is more efficient.

The second method was first introduced by David Huffman [7] and later referred as *Huffman Coding*. Huffman codes are created by analyzing the data set and assigning short bit stream to the datum (or pixel value) occurring most frequently. This algorithm attempts to create codes that minimize the average number of bits per image pixel. The beauty of Huffman Coding is the use of variable length codes. These codes have various lengths: the more frequently the value appears in the image, the shorter code will be assigned to represent it. As mentioned in [8], for complex image, Huffman coding alone will typically reduce the file size by 10% to 50%, which are equivalent to compression ratio 10:9 to 2:1. The compression ratio can be improved to 3:1 by preprocessing for irrelevant information removal. When Huffman coding is used for JPEG encoding, it is common to break an image into 8×8 blocks for encoding [9].

One of the most common compression algorithms used in computer images is *Lemple-Ziv-Welch Coding (LZW)* [10]. This lossless method can be found in several image file format, such as GIF, TIFF and overall PDF format. The LZW coding algorithm works by encoding strings of data. Each string of data is a sequence of grey values of contiguous pixels from an image. First of all, it produces an initial dictionary of some original patterns (or strings) with indexes. Then it starts to read the input pixel grey value from the beginning of an image and appends the grey values of continuous pixels to the string until it cannot find a match in the dictionary. The string consisting of the matched string and the next pixel value appended is recorded as a new pattern and saved in the dictionary. The index of the matched pattern (string) is encoded and the value of its succeeding pixel is recorded as the initial value for next pattern (string). The encoding process continues in this manner until comes to the end of input image. With LZW compression method, if the de-compression process uses the same initial dictionary as that for image compression, the dictionary itself is not necessary to be saved with the compressed file because it can be re-generated during the de-compression.

Another lossless compression method is called *Arithmetic Coding* [11, 12]. This method divides the original image into smaller sub-images and compresses each block with specific code word, which is normally a single floating point number between 0 and 1. Arithmetic Coding is a two-pass algorithm. The first pass called *modeling* computes the data frequency and generates a probability table. Based on this table, the more frequently occurring pixel values are assigned wider ranges in the interval and are requiring fewer bits to represent them. On the other hand, the less likely characters are assigned more narrow ranges and are requiring more bits. The second pass called *coding* does the actual compression.

Lossy methods

A simple method used in lossy image compression, called *Gray-level Run-Length Coding* [13], inherits the simple Run-Length Coding (RLC) for loseless image compression. It uses a smaller set of gray levels to represent the original image and applies the standard RLC technique. The 'Dynamic Window-Based RLC' is one of the most popular RLCs in practice. This algorithm relaxes the criterion of the runs being the same value and allows the runs to fall within a gray-level range, called the *dynamic window range*. This range is dynamic because it starts out larger than the actual gray-level window range, and maximum and minimum values are narrowed down to the actual range as each pixel value is encountered. Similar to the simple RLC, the image is encoded with two values, one for the run-length and the other for the approximate gray-level value of the run. This approximate value can be the average of

all the gray-level values in the run, or a value computed using a more complex method.

The *Block Truncation Coding (BTC)* [14, 15] method is popular for its low computational complexity and the preservation of edge and single pixel resolution. This method works by dividing image into small sub-images and then reducing the number of gray levels within each block. A quantizer that adapts to the original image statistics achieves this reduction. The levels for the quantizer are chosen to minimize a specified error criterion, and then all the pixel values within each sub-image are mapped to the quantized levels. There are many different BTC algorithms that have been defined using various types of quantization and error criteria, as well as various pre-processing and post-processing methods. Generally speaking, "*the more complicated algorithms provide better results but with a corresponding tradeoff of increasing computation time*" [8].

The third lossy compression method is *Vector Quantization* [16, 17]. Here '*Quantization*' means the process of representing a large, possibly infinite, set of values with a much smaller set. Generally, there are two types of quantization. One of these types is *scalar quantization*, which uses a single level value to represent a pixel in a sub-image as we discussed previously in the previous paragraph for BTC. The other type of vector quantization is *vector quantization*, which applies to every pixel on the entire 'sub-image'. Each vector corresponds to a sub-image, or a block. Vector Quantization is done by utilizing a *codebook*, which stores a fixed set of vectors, and then encodes the sub-image by using the index into the codebook. The toughest part during the process of vector quantization is to generate the codebook. In practice, the codebook is typically fulfilled by a training algorithm that finds a set of vectors that best represents the blocks in the images. The set of vectors is determined by optimizing some error criteria, where the error is defined as the sum of difference between the original sub-images and the resulting decompressed sub-images [8]. Some popular methods using vector quantization can be found in [16] and [18].

Instead of encoding a signal directly, *Differential Predictive Coding* [19] technique codes the difference between the signal itself and its prediction. That is to say, this method works by predicting the next pixel value based on the previous values and encoding the difference between the predicted value and the actual value. This technique takes advantage of the fact that adjacent pixels are highly correlated, which means that the difference between adjacent pixels is typically small. Because the difference is small, it will take only a small number of bits to represent. The use of a predictor allows us to further reduce the amount of information to be encoded. By using a simple prediction equation, we can estimate the next pixel value and then encode only the difference between the estimate and the actual value. For a higher compression ratio, a method called *Differential Pulse Code Modulation (DPCM)* can be applied. In DPCM coding, the difference signal is quantized and codeword is assigned to the quantized difference. There are two typical predictors classified in DPCM coding according to their dimensionality for still image compression. They are 1-D and 2-D predictors. DPCM with a 2-D predictor demonstrates better performance than a 1-D predictor since it utilizes more spatial correlation. A 2-D predictor uses not only horizontal correlation but also vertical correlation.

Transform Coding [20, 21] is a form of block coding done in the transform domain. The image is divided into blocks, or sub-images, and the transform is calculated for each block. Any transform such as linear transform can be used. It has been claimed that the discrete cosine transform (DCT) is optimal for most images [20]. After the transform is found, the transform coefficients are quantized and coded. The primary reason that this method is effective is because the transform of images efficiently puts most of the information into relatively few coefficients so that many of the high-frequency coefficients can be quantized to 0 (or eliminated completely). This type of transform is really just a special type of mapping that uses spatial frequency concepts as a basis for the mapping. For image compression, whole idea of mapping the original data into another mathematical space is to pack the information into as few coefficients as possible. An internationally recognized standard, JPEG standard, uses DCT as its basic coding scheme. This is the baseline of JPEG and is sufficient for many applications [22].

Hybrid methods

In image compression field, it is not uncommon to implement more than one method to achieve a better result. *Hybrid Methods*, as its name implies, use both the spatial domain and the transform domain. Model-based image compression, such as *Object-Based* and *Fractals* methods, can be considered as a hybrid method, and the transform used may be an object-based transform rather than a block-based one. Model-based compression works by finding models for objects within the image and using model parameters for the compressed file. The objects are often defined by lines or shapes (boundaries).

Advantages and disadvantages of current techniques

RLC scheme, the same as Gray-Level Run-Length Coding, is simple and fast but the compression efficiency depends on the type of image data encoded. A black-and-white image or images with solid backgrounds, like cartoons, can be encoded very well, due to the large amount of contiguous data that is all the same color. However, for those natural images or images without many repetitive symbols, RLC may even enlarge the file size.

The most outstanding character of Huffman Coding is the use of variable length code. The determination of code length depends on the analysis of data occurrence frequency. Consequently, Huffman Coding needs more time to proceed. The use of variable length code also means the process is not on the 'byte' boundary any more and any corrupted bit during compression may cause a fatal failure over the whole file.

When applying LZW, the codeword always uses more bits than the original data so that any single encoded value will expand the data size. This is always seen in the early stages of

compressing a data set with LZW. However, LZW compresses repetitive sequences of data well. Especially after a reasonable string table is built, compression improves dramatically. Compared with other image compression methods, LZW has the following advantages: it requires no pre-processing; the string table is built on the fly during compression and decompression, and it not necessary to be stored with the compressed data; the computation is reasonably fast and easy for both compression and decompression.

Arithmetic Coding requires the possibility of each value within the sub-image to be known exactly by pre-processing the image with a modeler. In Arithmetic Coding, the modeler and the coder can operate independently, which permits any degree of complexity in the modeler without any change to the coder. Each code is a single floating number, which may make the compression slower due to the complexity of computation.

Block Truncation Coding (BTC), Vector Quantization (VQ) and Transform Coding have the same first step: to divide the original image into smaller sub-images. Any of the VQ algorithms is computationally intensive during the encoding stage because of the creation of codebook. But the decoding is relatively quicker by merely pulling vectors out of the codebook and re-building the image. Therefore VQ is particularly suitable for some application that users do not care too much about the encoding process but do expect to view images quickly.

Differential Predictive Coding (DPC) has two processes: a predictor is first designed to predict the difference between two neighboring pixel values; then, a quantizer is optimized for the distribution of signal difference. It codes the difference rather than the pixel value itself, which makes DPC quite efficient.

Hybrid Method is simple a combination of any other compression methods. The compressed image via a method can be the input data for another compression process with different method. Based on the specific requirements, users can select any combination of compression methods to achieve the best result.

We summarize the various image compression schemes in Table 1 attached.

III. SPIRAL ARCHITECTURE

On Spiral Architecture, each (hexagonal) pixel is assigned a number called 'Spiral Address'. An example of a cluster with size of 7^2 and the corresponding addresses are shown in Fig. 2. The importance of the hexagonal representation is that it possesses special computational features that are pertinent to the vision process.

On Spiral Architecture, two algebraic operations have been defined, that are Spiral Addition and Spiral Multiplication [23] based on Spiral addresses. These two operations form two image transformations, which are translation of image and scaling rotation of image.

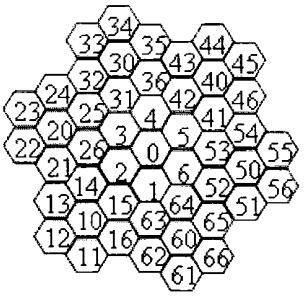
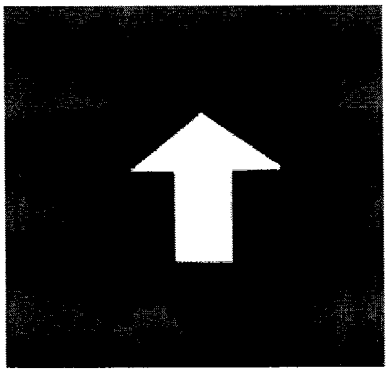
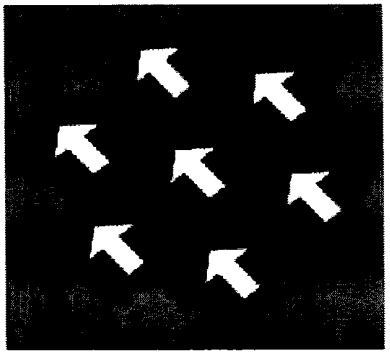


Fig. 2. A collection of 49 hexagonal pixels with assigned Spiral Addresses

In this paper, we will use Spiral Multiplication to achieve uniform image separation. Spiral Architecture improves image partitioning performance in terms of accuracy and speed. The operation of a Spiral Multiplication is fast as it can be converted to an addition operation as described in [23]. Spiral Multiplication improves the accuracy of image processing by the fact that it separates image into sub-images, of which each is similar to the original image. Fig. 3 demonstrates the result of separating (or partitioning) an image into seven sub-images using a Spiral Multiplication.



a) Original image



b) Result of image separation

Fig. 3. Image partition on Spiral Architecture

IV. IMAGE COMPRESSION ON SPIRAL ARCHITECTURE

In this section, we propose image compression schemes based on one-dimensional spiral addressing and the uniform image separation on Spiral Architecture (SA). All lossless and lossy image compression methods described in Section II will be examined based on SA. Then, hybrid methods (algorithms) on SA will be proposed for both lossless and lossy image compression.

Lossless image compression

Based on our research done in paper [4], the Spiral Architecture has better locality, i.e., adjacent pixels having closer grey values. It can be expected that RLC will work better along the spiral addressing direction as the length of each run is expected to be longer in general.

We apply Huffman coding to images represented on SA. Rather than breaking an image into 8×8 blocks, we break each image into blocks of $7^2 = 49$ hexagonal pixels with continuous spiral addresses.

As mentioned above, image separation on SA creates sub-images which are very similar. It is expected to see that the patterns in each sub-image are close to those in other sub-images. Hence, LZW compression method will work well after image separation on SA.

Note that Arithmetic Coding requires image separation. On the other hand, image separation can be easily done on SA for as many similar sub-images as required. Hence, it is not doubtful that SA is a perfect structure for arithmetic coding.

Lossy image compression

Similar to RLC for lossless compression, *Gray-level Run-Length Coding* will receive its better compression ratio when it applies to SA.

When working with BTC, SA will make it easier to determine the quantizer levels in each sub-image because its similarity to the original image.

The uniform image separation on SA also helps the training process for vector quantization of a sub-image. The reason is that the similar sub-images should also produce similar vectors representing the sub-images. A neural network can be applied for the training process.

Because the difference between any two sub-images produced by an image separation on SA is very small, Differential Predictive Coding will well fit into SA.

In a most recently published paper [21], a simple transform of polynomial axy+bx+cy+d was introduced. This can be applied to Transform Coding in Spiral Architecture after uniform image separation.

Hybrid image compression

We list some possible hybrid image compression methods on SA as follows.

Step 1. Uniformly separate an image on Spiral Architecture into similar blocks of 49 pixels.

Step 2. Use either RLC, Huffman Coding, Arithmetric Coding or BTC for coding an image block.

Step 3. Use either LZW, Differential Predictive Coding or Transform Coding to encode other image blocks.

Many more hybrid compression methods can be created based on the features of Spiral Architecture.

V. CONCLUSION

To answer how SA can improve image compression, this paper has presented possible approaches to image compression on Spiral Architecture. The development of a JPEG-like compression standard (or format) based on Spiral Architecture will be sought after the implementation of the image compression schemes discussed in this paper. Video (moving) image compression based on Spiral Architecture is another direction of our future work.

REFERENCES

[1] P. Sheridan, T. Hintz. and W. Moore, *Spiral Architecture in Machine Vision*, Proceedings of Australian Occam and Transputer Conference, Amesterdam, 1991.

[2] E. Schwartz, Computational Anatomy and Functional Architecture of Striate Cortex: A Spatial Mapping Approach to Perceptual Coding, Vision Research 20, pp.645-669, 1980.

[3] P. Sheridan, T. Hintz and D. Alexander, *Pseudo-invariant image transformations on a hexagonal lattice*, Image and Vision Computing, **18**(2000), pp.907-917.

[4] T. Hintz and Q. Wu, *Image Compression on Spiral Architecture*, Proceedings of The International Conference on Imaging Science, Systems and Technology, Las Vegas, Nevada, USA, pp. 201-204, 2003.

[5] Qiang Wu, Xiangjian He and Tom Hintz, *Preliminary Image Compression Research Using Uniform Image Partitioning on Spiral Architecture*, ICITA04, Harbin, China. January 2004, to appear.

[6] B. B. Aprs, *Binary Image Compression, in Image Transmission Techniques*, W. K. Pratt(Ed.), New York: Academic Press, 1979.

[7] D. A. Huffman, *A Method for the Construction of Minimum-redundancy Codes*, Proc. IRE, 40, pp.1098-1101, 1952.

[8] S. E. Umbaugh, *Computer Vision and Image Processing: a Practical Approach Using CVIPtools*, Prentice Hall, 1996.

[9] P. Symes, *Compression: Fundamental Compression Techniques and a Overview of the JPEG and MPEG Compression Systems*, McGraw-Hill, New York, 1998.

[10] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures*, Kluwer Academic Publishers, 1997.

[11] C. D. Hardin and S. Zabele, *Arithmetic coding for lossless and loss-inducing image compression*, Multidimensional Signal Processing Workshop, 1989., Sixth , 6-8 Sep 1989.

[12] P. G. Howard and J. S. Vitter, *Arithmetic coding for data compression Proceedings of the IEEE*, Volume: 82 Issue: 6, pp. 857 -865, Jun 1994.

[13] K. Sayood, *Introduction to Data Compression*, Academic Press, 2nd Edition, 2000.

[14] E. J. Delp and O. R. Mitchell, *Image Compression Using Block Truncation Coding*, IEEE Trans. Commun. COM-27 (1979), p.1335-1342.

[15] Chung-Woei Chao, Chaur-Heh Hsieh and Po-Ching Lu, *Image compression using modified block truncation coding algorithm*, Source SIGNAL PROCESSING IMAGE COMMUNICATION [0923-5965] 1998 Vol.12.

[16] Y. Linde, A. Buzo and R. M. Gray, *An Algorithm for Vector Quantizer Design*, IEEE Trans. Commun., vol. COM-28, p.84-95, Jan. 1980.

[17] N. M. Nasrabadi and R. A. King, *Image Coding Using Vector Quantization : A Review*, IEEE Trans. Commnun., vol. 36, p957-971, Aug. 1988

[18] Y. H. Shin and C. C. Lu, *Image compression using vector quantization and artificial neural networks*, IEEE International Conference on, 13-16 Oct 1991.

[19] A. Habibi, *Comparison of nth-order DPCM encoder with linear transformations and block quantization techniques*, IEEE Trans. Commun. Technol., COM-19(6), 948-956, 1971.

[20] T. A. Welch, *A Technique fro High-Performance Data Compression*, IEEE Computer, page 8-19, June 1984.