Digital Video Compression Using DCT-Based Iterated Function System (IFS)

Nevart A. Minas , Faten H. Al-Qadhee
Computer Systems, Technical Institute, Northern Technical University, Foundation of Technical Education, Iraq

Abstract

Large video files processing involves a huge volume of data. The codec, storage systems and network needs resource utilization, so it becomes important to minimize the used memory space and time to distribute these videos over the Internet using compression techniques. Fractal image and video compression falls under the category of lossy compression. It gives best results when used for natural images.

This paper presents an efficient method to compress an AVI (Audio Video Interleaved) file with fractal video compression(FVC). The video first is separated into a sequence of frames that are a color bitmap images, then images are transformed from RGB color space to Luminance/Chrominance components (YIQ) color space; each of these components is compressed alone with Enhanced Partition Iterated Function System (EPIFS), then fractal codes are saved.

The classical IFS suffers from a very long encoding time that needed to find the best matching for each range block when compared with the domain image blocks. In this work, the (FVC) is enhanced by enhancing the IFS of the fractal image compression using a classification scheme based on the Discrete Cosine Transform (DCT). Experimentation is performed by considering different block sizes and jump steps to reduce number of the tested domain blocks. Results shows a significant reduction in the encoding time with good quality and high compression ratio for different video files.

Keywords: IFS, PIFS, Fractal Video Compression , FIC and DCT.

1. Introduction

Data rates have to be reduced in order to transmit video data over networks. The main idea of video compression is to exploit redundancies that are present in the video. The Audio Video Interleave file format was originally developed by Microsoft for Intel. The AVI format is now marketed as “Video for Windows”. In general, AVI files contain multiple streams of different types of data. Most AVI sequences will use both audio and video streams a standard package to allow its simultaneous playback[1].

Fractal image compression (FIC), is a loosy compression method based on the self similarity property of an image, some image parts called the range block are similar to a large part of another image called domain block. Barnsley and Jacquin[2][3], were the first to used the idea of Iterated Function System(IFS). IFS is a collection of contractive affine transformations, and each set of these transformations defines a unique image or what is called fractal[4]. Jacquin proposed a new fractal image compression method based on image block, and the method can conduct automatically without manual intervention[5], He improved the IFS proposing the partitioned iterated function system (PIFS), where the image is partitioned into number of ranges that are non-overlapping, and range-wise IFS are found and can be automatically done. Currently, fractal image compression has got extensive attention because of its novel ideas, high compression ratio, besides it can be applied to audio and video compression[6]. Different techniques like DCT, DWT, etc. are used reduce the PIFS long encoding time. The Discrete Cosine Transform (DCT) is an example of transform coding. The DCT coefficients are all real numbers unlike the Fourier Transform[7]. DCT technique is most commonly used in the compression of the images. Most of the visually significant information about the image is concentrated in a few coefficients of the Discrete Cosine Transform[8].

2. Partitioned Iterated Function System (PIFS)

The scheme of PIFS implies partitioning the image into non-overlapped blocks; these blocks are assembled in range pool. While, the extracted blocks from the down-sampled copy of the image are put in the domain pool[9]. The sizes and shapes selection for image cells depends on several factors. Small image blocks less than (4x4) are easy to analyze and to classify geometrically[10]. The IFS encoding of each range block implies choosing the most resemble domain block listed in the domain pool, and approximating the range by linearly mapping (affine transform) the selected domain block[11].

Fractal algorithms convert these parts into mathematical data called fractal codes. In a given input color image, similar blocks are identified using FIC (i.e. the matched domain blocks for each range block in an image). Given two blocks of pixel intensities, first from the domain pool (D_i), the second from range pool *(R_i)*, and using metric optimal values s_k for contrast setting and offset o_k , brightness setting are computed as following to minimize equation (1):

$$E = \min(\|r_i - (s_k d_k + o_k)\| k = 1,2,....7 \ldots\ldots(1)$$

Where s_k and o_k are calculated as:

$$s_k = \frac{\left(n \sum_{i=0}^{n} d_i r_i - \sum_{i=0}^{n} d_i \sum_{i=0}^{n} r_i - \right)}{\left(n \sum_{i=0}^{n} d_i^2 - (\sum_{i=0}^{n} a_i)^2 \right)} \quad \ldots\ldots..(2)$$

$$o_k = \frac{1}{n}\left[\sum_{i=0}^{n} r_i - s_k \sum_{i=0}^{n} d_i\right] \text{........(3)}$$

Before comparing domain blocks, the range image is down sampled by taking average of four neighboring pixels to make domain blocks equal to size of range blocks. Then each domain block is compared with the range block by trying eight different affine transformations[12]. Compare each *R* block with all *D* blocks in domain pool, and obtain the most similar *D* block. For each *R* block, record the corresponding compression affine transformation *W.* All compression affine transformations constitute the whole image's fractal code[5].

For each domain block contrast scaling and brightness shift are calculated by using equation (2) & (3), the domain block which minimizes equation (1) will be the best matched domain block. The fractal affine transformation for domain block is expressed as:

$$W\begin{bmatrix} x \\ y \\ d_k \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ 0 & 0 & s_k \end{bmatrix}\begin{bmatrix} x \\ y \\ d_k \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ o_k \end{bmatrix} \text{.........(4)}$$

In practice, tx, ty, sk, ok , and k are encoded using 8, 8, 5, 7 and 3 bits, respectively, that are regarded as the compression code of r_i[12].

3. The Proposed Video Compression

Compression basically means reducing the image data with a little affection on their quality. Often more than 50% compression can be achieved without visible difference in quality. Because of the unique features of the fractal coding, this paper, proposed Fractal Video Compression method using PIFS used in fractal coding and DCT transform to compress the AVI videos files. As mentioned above, the main disadvantage of classical IFS scheme is the long time needed to encode the images which lead to slow video compression. So an efficient enhancements have been made on the IFS searching scheme to speed up the encoding time with a very little effect the image quality.

In the encoding stage, the AVI video file is loaded and the sequence of frames are extracted as a bitmap image files. These images are converted from RGB to YIQ color components (Y is the luminance component, I and Q are the chromatic components of the image). Each of these components which is considered as the range image, is compressed using the enhanced partitioned iterated function system (EPIFS) encoder separately. The enhanced encoder consists of a set of transforms applied to create the domain image pool from the range image by down-sampling it by 2 because the domain preferred to have block patterns similar to the range blocks, although it is possible to use any other image as domain pool. The range image is partitioned to non overlapped blocks of length (*L*) and partition the domain image to overlapped blocks of the same length to increase the search region.

A set of contractive affine transforms are applied to each range block to find the most similar domain block. The enhancement is done to the classical IFS uses a DCT based descriptor(*R*) to classify the domain and range blocks. The descriptor is computed for the range and domain blocks depending on the energy compaction property of the low frequency coefficients of the DCT transform. The energy decreases when moving to high frequency part of the DCT domain, this part of coefficients holds the minimal information. The even coefficients like T_{20} & T_{02} are invariant to block reflection and they will be the same as without reflection. For this reason, they are ignored, so only the coefficients T_{10} or T_{01} are enough to be used to find DCT based descriptor from the following equations:

$$T_{10} = \sum_{y=0}^{L-1}\sum_{x=0}^{L-1} t(x,y)\cos\left(\frac{\pi}{2L}(2x+1)\right) \text{............(6)}$$

$$T_{01} = \sum_{y=0}^{L-1}\sum_{x=0}^{L-1} t(x,y)\cos\left(\frac{\pi}{2L}(2y+1)\right) \text{............(7)..}$$

where,

$t(x,y)$ is the domain or range element , *L* is the length of domain and range element.

R descriptor is calculated based on the ratio difference of the low DCT coefficients into categories or classes. Number of classes is determined by the parameter (*Nc*). The *R*-descriptor is calculated from the following equation:

$$R_n = \begin{cases} Nc * \dfrac{T_{01}}{T_{10}} & if T_{10} > 0 \\ Nc * \dfrac{T_{10}}{T_{01}} & if T_{01} > T_{10} \end{cases} \text{...... (8)}$$

The domain blocks are sorted according to their *R*-descriptor values in an ascending order to reduce the matching time for each range block, and instead of testing all domain blocks for the best match, only the domain blocks with same *R* values are tested. Another enhancement is done by the Boolean relation of the low DCT coefficients that is used to predict the isometric(or symmetry) operation to avoid testing all the eight symmetry cases, this reduces the searching time to 1/8. The matched domain block parameters are: scale(s), offset(o), symmetry, and domain location, as mentioned above, these parameters are recorded for each range block.

To increase the compression gain at the encoding stage, the best matched domain block parameters is quantized and then Differential Pulse Code Modulation (DPCM) is applied to them. The DPCM encoding records the difference of every two adjacent values (except the first value) to reduce their values before saving them into the compressed video file. The IFS parameters of the encoded video frames are recorded sequentially into the compressed file.

The proposed video compression encoding steps are explained in the following algorithm:

Step 1: Load the AVI file

Step2: Extract frames from the video into a series of frames (color bitmap images).

Step3: For each extracted frame do the following:

Step 4: Convert the RGB image into YIQ components.

Step 5: For each component (Y, I and Q) do the following:

- Partition the range image into non-overlapping blocks with block length=n.
- Down sample the range image by 2 to create the domain pool blocks.
- Partition the domain image into overlapping blocks with block length=n.
- find the R-descriptor for all domain blocks from the DCT low coefficients.
- Sort the domain pool blocks according the R descriptor.
- Find R-descriptor for the selected range block and compare it with domain blocks(with suitable jump step) that have the similar R-descriptor with the predicted isometric state till finding the best match with minimum threshold.
- Quantize the best match parameters, and apply DPCM encoding.
- Store the transformation parameters of the matching.

End for (of step5)

End for (of step 3)

End encoding

Decompression each frame is simpler to the coding process using a reverse steps but it is faster. The fractal codes stored in the compressed AVI file of each frame components(Y, I, and Q) are decoded using the inverse fractal coding, Using the stored transform data, the reconstruction of the image is obtained by iterating the mapping, typically ten iterations are enough to obtain the final image component of YIQ color space, these components then are transformed back to RGB color space image. The resulting decoded images are combined together as a frames to create the reconstructed AVI video file. These videos runs at rate of (30) frame/second.

4. Experimental Results

The proposed approach is suggested for enhancing the PIFS compression process using the DCT transform to reduce the encoding time. In order to test the performance of the proposed algorithm, experimental results are carried out on the four AVI video files (Kid, Bird, Dance and Car) shown in table (1), each has different number of frames and sizes. The video frames are extracted as uncompressed bitmap images (BMP). Performance parameters (size of the video before and after the compression, average PSNR, Compression ratio, encoding and decoding time) were registered.

The algorithm is used to compress each video many times with different fractal parameters like block sizes and jump step.

Table (2) compares the videos depending on the block size (4,5, and 8) which significantly increases the compression ratio and quality when increased, but also it increases the encoding time. The encoding time for each video frame individually can be computed approximately by dividing the encoding time in the table (2 &3) by the number of frames in that video. For each frame in these videos the encoding time of the individual frame is about (2-3) seconds, and this value is good comparing to previous researches.

In table(3) a comparison is made to see the effect of the jump step which is clearly reduced the encoding time to the half when changed from 2 to 3, but the degradation in the decoding time, quality and compression ratio are not mentioned. The coding time for individual frames is reduced when jump step is 3, as an example the encoding time for the first video frames is reduced from (2.6) at jump step is 2 is reduced to (1.6) at jump step 3, and this time can be further reduced if the jump step increased to 4 with little degradation in quality that may not appear in the reconstructed video motion. In common all results of the reconstructed videos shows a good PSNR and very high compression ratios (above 90%) of the original video size, in addition to reducing encoding and decoding time.

5. Conclusions

The video compression test results proved that the proposed scheme was very efficient in compressing videos, and from the results of the tested parameters, conclusions could be summarized as the following:

1. The sizes of the AVI videos were the same after the reconstruction with very good quality because of using the high quality bitmap images and the efficient IFS method.
2. Using the DCT transform in the affine transformation gives low encoding time because of prediction of the symmetry cases to reducing it to one, and classifying tested domain blocks.
3. Low encoding time is obtained by increasing the jump step which helped to avoid testing all domain blocks to find the attractive, while increasing the block size was inversely effected on the encoding time by increasing the search time for bigger blocks.
4. The decoding time was little affected in all cases because it only use the stored transform data to reconstruct the image.
5. The quality was little affected when increasing the jump step, and does not appear in the video motion. Also using a very big jump step is not preferred because it will increase the time needed to find the attractive.
6. Compression ratio is proportional to the block size, the maximum CR obtained was when block length is 8, but it is not recommended to increase the block size more than 8 to avoid increasing time.

ISSN: 1813 – 1662 (Print)
E-ISSN: 2415 – 1726 (On Line)

Table (1) Selected frame samples from used videos before & after compression

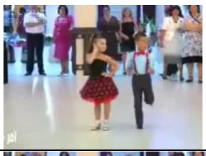
	Video Frames examples before and after compression				
Video-1 (234-Frame)	**10**	**30**	**60**	**90**	**110**
Reconstru-cted					
Video-2 (73-rame)	**8**	**15**	**35**	**55**	**70**
Reconstru-cted					
Video-3 (200-rame)	**1**	**40**	**90**	**175** 	**200**
Reconstru-cted					
Video-4 (25-rame)	**1**	**7**	**13**	**19**	**25**
Reconstru-cted					

Table (2) The Tested parameters depending on block size and jump step is 2

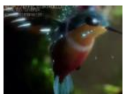
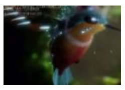
Video	Original Size(MB) And(frame Dim)	No. of Frames in	Block size	Average PSNR(dB) of the video	Compressed Size of the video	CR Of the video	E-time (Sec)
	90.4 (544x248)	234	4	41.12	9.15MB	89.88	521.8
			5	40.12	5.95 MB	93.52	611.30
			8	36.80	2.34	97.41	687.70
	30 (444x324)	73	4	32	3.14 MB	89.5	232.86
			5	31.55	2.04 MB	93.2	261.71
			8	27.33	836 KB	97.21	290.73
	16.1 (400x224)	63	4	32.54	9.15 MB	43.17	155.96
			5	29.69	1.13 MB	92.98	161.79
			8	25.94	452KB	97.19	148.84
	5.73 (320x240)	25	4	30.80	560 K	90.23	54.16
			5	28.73	357 K	93.77	57.76
			8	25.90	139 K	97.57	54.74

Table (3) The Tested parameters depending on Jump step and Block size is 5

Video	Jump step	Average PSNR(dB)	Encoding Time(s)	Decoding Time(s)	CR
	2	41.12	611.30	276	89.88
	3	40.38	386.33	257.07	90.28
	2	31.55	261.71	87.58	89.53
	3	30.87	171.37	89.39	90.03
	2	32.54	155.96	44.69	43.17
	3	31.63	97.91	46.61	43.77
	2	30.80	54.16	15.52	90.23
	3	30.10	39.93	15.60	90.80

7. References

[1] Hashim T. Ashawq, Ali H. Yossra, & Ghazou S. Susan., **Developed Method of Information Hiding in Video AVI File Based on Hybrid Encryption and Steganography**, Eng. & Tech. Journal, Vol.29, No.2, 2011.

[2] Barnsley M. F. and Jacquin A., **Application of recurrent iterated function systems to images**, Visual Comm. and Image Proc. 88, Third in a Series, vol. 1001, pp. 122-131, Oct. 1988.

[3] Jacquin A., **Fractal image coding based on a theory of iterated contractive image**

transformations, in Proc. SPIE Visual Comm. and Image Proc.90, Fifth Series, Vol. 1360 pp. 227-239, Sep. 1990.

[4] Brijmohan Y. and Mnene S. H., **Video Compression for Very Low Bit -Rate Communications Using Fractal and Wavelet Techniques**, e-book browse, No 19, 2010.

[5] Hai Wang, **Fast Image Fractal Compression with Graph-Based Image Segmentation Algorithm**, International Journal of Graphics Vol. 1, No.1, Nov. 2010

[6] Shrirame W., Asutkar G. M., **Compressed Video Data Transmission over Wireless Network**, International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056 Vol. 2, Issue 04 , Jul. 2015.

[7] Harjeetpal S. and Sharma S., **Hybrid Image Compression Using DWT, DCT & Huffman Encoding Techniques**, International Journal of Emerging Technology and Advanced Engineering, Vol. 2, Oct. 2012.

[8] Sharmila K., Kuppusamy K., **An Efficient Image Compression Method using DCT, Fractal and Run Length Encoding Techniques**, International Journal of Engineering Trends and Technology (IJETT) Vol.13 No. 6 , Jul. 2014.

[9] Y. Fisher, **Fractal Image Compression**, ACM SIG-GRAPH Course Notes, 1992.

[10] Jacquin, Arnaud E., **Image coding based on a fractal theory of iterated contractive image transformations**, IEEE Transactions on Image Processing, vol.1, no.1, pp.18-30,1992.

[11] George L. E., **Fast IFS Coding for Zero-Mean Image Blocks**, Iraqi Journal of Science, Vol 47, No.1, 2006.

[12] Kodgule U. B., Sonkamble B. A., **Discrete Wavelet Transform based Image Compression using Parallel Approach**, International Journal of Computer Applications, Vol. 122, No.16, Jul. 2015.

كبس الملفات الفيديوية الرقمية باستخدام نظام الدالة المتكرر (IFS) مبنيا على محول الجيب تمام (DCT)

نفارت الياس يوسف ميناس ، فاتن حسن القاضي

قسم تقنيات الحاسوب ، المعهد التقني كركوك ، الجامعة التقنية الشمالية ، هيئة التعليم التقني –العراق

الملخص

تتطلب معالجة الملفات الفيديوية الكبيرة حجم كبير من البيانات، لذا فان الشفرات، انظمة التخزين والشبكات تحتاج توفير مستويات عالية من المصادر، لذلك اصبح من الضروري تقليل المساحة المطلوبة لخزن البيانات وكذلك الوقت المطلوب لنشر ملفات الفيديو عبر الانترنيت باستخدام تقنيات الكبس. ان كبس الصور والفيديو باستخدام الكبس الكسوري يصنف ضمن الكبس المسبب لضياع او الفقد للبيانات (Lossy) يعطي نتائج افضل عندما يستخدم للصور الطبيعية.

يقدم هذا البحث نظرية كفؤة لكبس ملف فيديو رقمي نوع AVI (تداخل الصوت والفيديو) باستخدام طريقة كبس الفيديو الكسوري (Fractal Video Compression). المرحلة الاولى من البحث تضمنت فك الفيديو الى سلسلة من الاطارات (Frames) والتي هي عبارة عن صورة ملونة نقطية (Bitmap)؛ ثم يتم تحويل هذه الصور من نظام الالوان (RGB) الى نظام مكونات الانارة والتلون (YIQ)، يتم ضغط كل من هذة المكونات بشكل منفرد باستخدام نظام الدالة الكسورية المحسنة (Enhanced Partitioned Iterated Function System) ثم خزن العناصر الكسورية. ان الطريقة التقليدية لنظام الدالة الكسورية المتكرر (Iterated Function System) تعاني من مشكلة طول وقت الكبس المطلوب لايجاد التطابق الامثل عند لمقارنة كل بلوك من صورة المصدر مع بلوكات صورة الهدف.

في هذا العمل تم تطوير طريقة كبس الفيديو الكسوري (FVC) بتطوير نظام الدالة الكسورية (IFS) المستخدمة في الكبس الكسوري باعتماد نظام تصنيف مبني على اساس محول الجيب تمام (DCT). تم اجراء اختبارات بأعتماد احجام وخطوات قفز مختلفة للبلوكات لتقليل عدد بلوكات الهدف التي يتم مطابقتها. النتائج اثبتت تقليل ملحوظ في الوقت المستغرق للكبس، كفاءة عالية لملفات الفيديو المسترجعة ونسبة كبس عالية لمختلف الملفات الفيديوية.