Volume 3, Issue 6, June 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Speeding up Audio Fractal Compression

Dr. Salih M. Ali
Baghdad University/College of Science/Iraq

Dr. Loay E. George
Baghdad University/College of Science/Iraq

Hazeem B. Taher
Thi-Qar University/College of Science/Iraq

Abstract-- In this paper, a simple approach to speed up the audio fractal compression is introduced. Firstly, an improved mechanism for the traditional fractal is developed to reduce the coding time. It reduces the encoding time about 70%. Depending on dividing the hole data list to sub segments, and each segment is treated separately. The adopted Affine mapping between domain and range block is that based on scaling the zero mean values of both the range and domain sample. Also, during the domain pool search cycle a stopping criteria based on monitoring the affine approximation error is adopted to reduce the encoding time. The MSE and PSNR are used as fidelity criteria during the test phase.

Keywords-Compression, Audio, Fractal, Fractal Audio Compression, Tilling.

I. Introduction

Perhaps the most widespread applications of digital speech processing technology occur in the areas of digital transmission and storage of speech signals. In these areas the centrality of the digital representation is obvious, since the goal is to compress the digital waveform representation of speech into a lower bit rate representation [1].

One of the most important characteristics of fractal data coding is its asymmetrical property of encoding and decoding processing [2]. This is the main reason that fractal is not standard compression method.

This paper adopts an improved algorithm to speed up the fractal audio compression (FAC). It is based on applying audio contiguous framing based method on method on wave data (i.e., via tilling). Here the audio wave data is divided into sub waves, then applying fractal coding on each sub wave separately. In this project a wave file loaded and the audio data is extracted from the file to be fed to the fractal system.

II. Audio Fractal Compression

The first automated fractal coding algorithm is based on partitioned iterated function system (PIFS); it was developed by Jacquin [3]. AFC is a relatively recent audio compression method which exploits the existing similarities in different parts of audio. The basic idea is to represent the audio blocks using IFS coding set; and these sets are considered as good fractal representation (or transform) that leads to fixed point reconstruct. Therefore, the input audio can be represented by a series of IFS, which is basically based on affine transformation [4].

The basic idea of PIFS is partitioning the data into non overlapping range blocks. For every range block a similar but larger domain block is found. There are many ways to partition data. The way used in this paper is a fixed size partitioning scheme, because it requires less computational time than the other.

According to the improvements were performed on the affine mapping equations, a new set of affine equations were used and found more stable. L.E.George [5], had replaced the offset parameter by applying blocks averaging. So, the contractive affine approximation of zero-mean blocks will become [5]:

$$r' = s(d_i - \bar{d}) + \bar{r} \qquad (1)$$

Where,

$$s = \begin{cases} \frac{1}{m}\sum_{i=0}^{m-1} d_i r_i - \bar{d}\bar{r} & if\ \sigma_d^2 > 0 \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

$$\chi^2 = \sigma_r^2 + s\left[s\sigma_d^2 + \ 2\bar{d}\bar{r} - \ \frac{2}{m}\sum_{i=0}^{m-1} d_i r_i \right] \qquad (3)$$

And

$$\bar{r} = \frac{1}{m}\sum_{i=0}^{m-1} r_i \qquad (4)$$

$$\bar{d} = \frac{1}{m}\sum_{i=0}^{m-1} d_i \tag{5}$$

$$\sigma_d^2 = \frac{1}{m}\sum_{i=0}^{m-1} d_i^{\,2} - \bar{d}^{\,2} \tag{6}$$

$$\sigma_r^2 = \frac{1}{m}\sum_{i=0}^{m-1} r_i^2 \tag{7}$$

Where, r_i and d_i are the i^{th} samples of the approximated range block and mapped domain blocks, respectively. $\bar{r}$ and $\bar{d}$ are the average values of the range and domain blocks, respectively. σ_r^2 and σ_d^2 are the variance values of the range and domain blocks, respectively. m is the number of samples in each block. Here the fractal parameters are the scale (s) and range average ($\bar{r}$); where the latter is used instead of the conventional offset coefficient in traditional IFS mapping equation.

The scenario of domain search is to check each domain block and determine IFS parameter that minimize the error between the checked domain block with range block, the matching is continued over all the domain blocks till finding the domain block whose difference error (X^2) with range block is the minimum in comparison with errors registered by other domain blocks. Each domain is subjected to some isometric (symmetry) to get different symmetry state for each domain block, and then the transformed domain block is considered as individual domain block, which should be matched with range blocks as a separate or individual case.

To reduce the matching time the minimum permissible error value is use as a stopping condition threshold, which ending the matching process if the value of (X^2) computed from equation (3) is less than the minerror value (threshold) and then take the IFS parameter for this domain as best matching case.

An improved scheme for speeding up fractal audio compression is presented and demonstrated in this paper; it depends on the principle of signal tiling, and each tile could treated separately [6].

The audio fractal compression system consists of two main stages encoding and decoding each of them is applying by different algorithm. Figure (1) shows the steps of encoding stage.

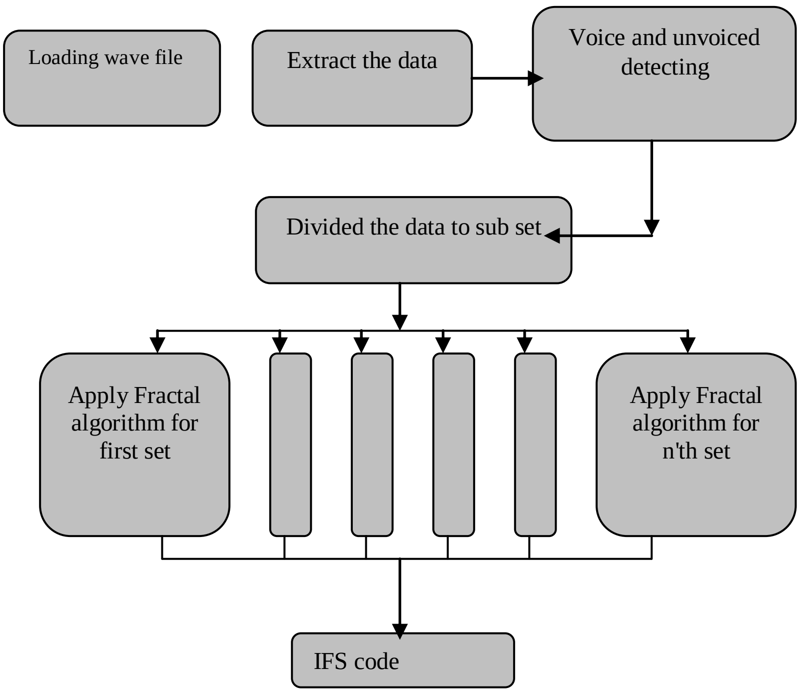


Fig. 1 Encoding stage.

III. Compression Stage

The encoding unit consists of a number of modules, which are together responsible for reducing the size of the desired audio data and generates the compressed file. The main step of the encoding unit is listed bellow [7]:

1. Loading the wave file: The specifications of the used input audio waveform data in this work are: 8-bits sample length, and mono (i.e. single channel).
2. Extract the audio data, wave (), (split the data from file header).
3. Divide the audio data in small blocks, and for each block check weather it contain information or it is a silent block), this a accomplished by implementing the following.

Input: *wave data buffer bwav (wavesize-1)*

Output: *A Flag indicate wether the block is voiced or unvoiced.*

For each block compute the number of sign exchanges between its adjacent samples. If the number of sign exchanges >0 then

Set unvoiced buffer () **=1**

else

Set unvoiced buffer () **=0**

4. For each tile do the following:
    a) Partition the whole tile into non overlapping blocks.
    b) For each blocks do the following:
        i. Classify the blocks types into voiced or unvoiced blocks.
        ii. Repeat step (i) for whole un-overlapped blocks of the audio signal, the unvoiced blocks are approximated as zero-average blocks using run-length encoding.
5. For voice class do the following steps:
    a) Construct the Range pool by partitioning the tile data into non-overlapped blocks:

Input: *wavetile data wav (wavesize-1; wavesize is the size of the tilewave data buffer is the size of the range block.*

Output: *wave partitioned into R_b range blocks (R_b is the number of range blocks)*

Set R_b = wavesize/b

For i=1 to R_b

For j=1 to b

Set $r_i(j) = wav((i-1)*b + j)$

Next j

Next i

End

    b) Construct the domain pool by down sampling, by 2, the whole tile data:

Input: *wave data (bwav (wavesize-1))*

Output: *1D array called domain pool, its size is domainsize, which is half size of the range pool.*

Set Domainsize= wavesize\2-1

For Pd=0 to domain size

Set Pr=Pd+Pd

Set Domain (Pd) = (round (wav (pr) +wav (pr+1))\2)

End

    c) Compute the parameters d and σ_d^2 (for all domain blocks) and put them in array.
    d) For each range block search the domain pool, and match the Range block with each domain block using affine transform equations (2 and 3). Find the best matched domain block, and register the corresponding affine coefficients (scale, symmetry, and position), of this domain block.

Input: *Range blocks with number No. Range Blocks*

Output: *scale, posI, Sym of each range block.*

```
For i=0 to No. Range Blocks
Rb is the ith Range block.
Compute range parameters using equation (2 and 7).
Set MinError= 9E+19.
For j=0 to no.domainblocks
Db is the j^th Domain block.
Set Pd=0:srd1=0:srd2=0
For m=0 to blocksize-1
Set srd1=srd1+domain (m+Pd)*rangeblock (m)
Set srd2=srd2+domain (m+Pd)*rangeblock (blocksize-1-m)
End m
For sym=0 to 1
If sym=0 then
Set srd=srd1
        Else
Set srd=srd2
Compute scale values of Rb and Db using equation (2).
Quantize and dequantize scale value.
Compute the distortion error (chi) between Rb and Rd using equation (3).
If chi< minerror the
End
Set Pd=Pd+jumpstep
```

End

6. For each range block store IFS parameter (scale, position, mean of range block and symmetry).

IV. DECOMPRESSION STAGE

In the decoding unit the decoded wave data is transformed into a set of PIFS codes while, in the decoding unit these PIFS codes are used to iteratively reconstruct the wave data.

Decoding can be done by starting with an arbitrary initialized range pool, then the domain pool is generated by down sampling the range pool, estimating the range blocks of a new source from the contracted domain blocks, and iterating the above last two steps until attractor state is reached, or nearly so. If the range block estimators are good enough then the fixed point source will be much like the original source.

V. EXPERIMENTAL RESULT

This section presents some of experimental results obtained for different six audio files. The wave file audio stream is 11025 Hz, 8 bits/sample wav file. To show the performance, some testing results were arbitrarily selected. The MSE and PSNR and time are listed in the following tables, figures and waveform.

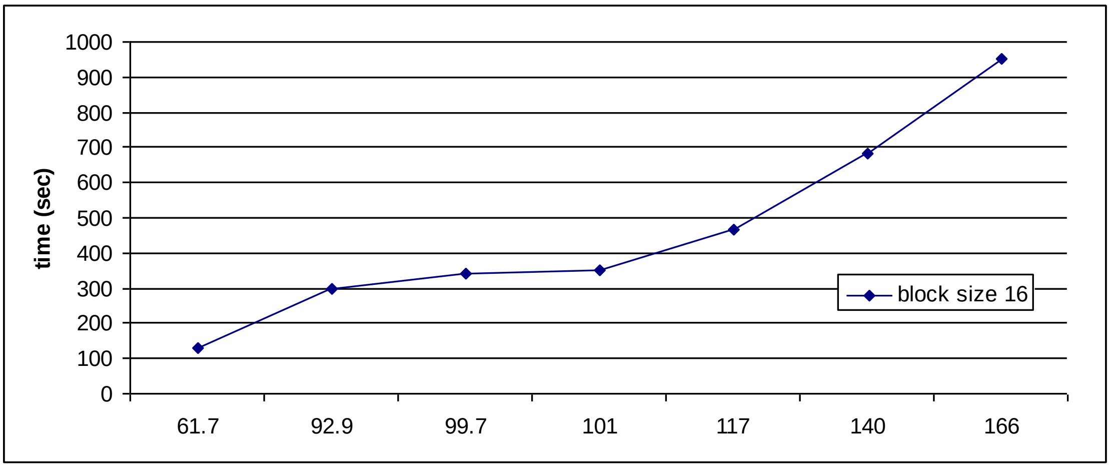


Fig. 2 The relation ship between time and data size

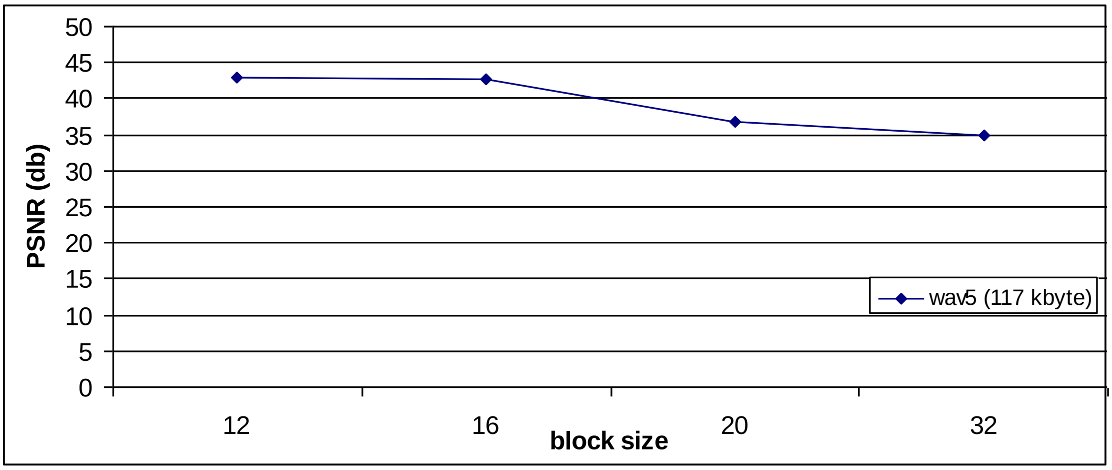


Fig. 3 The relationship between PSNR and block size

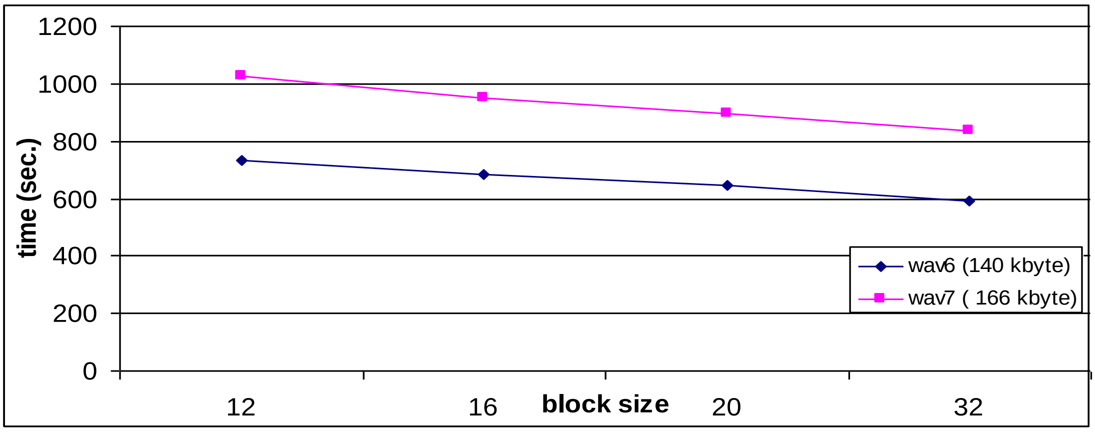


Fig. 4 The relationship between coding time and block size.

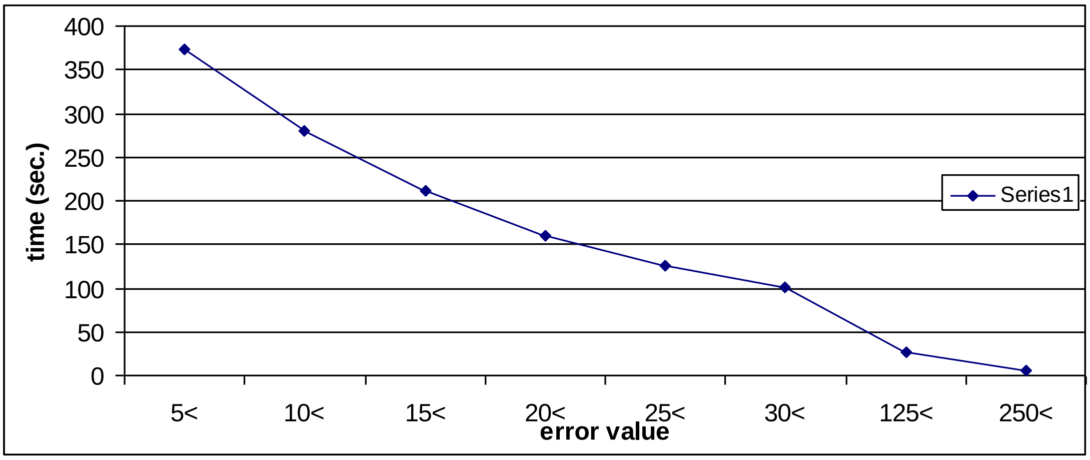


Fig. 5 The relationship between coding time and error value.

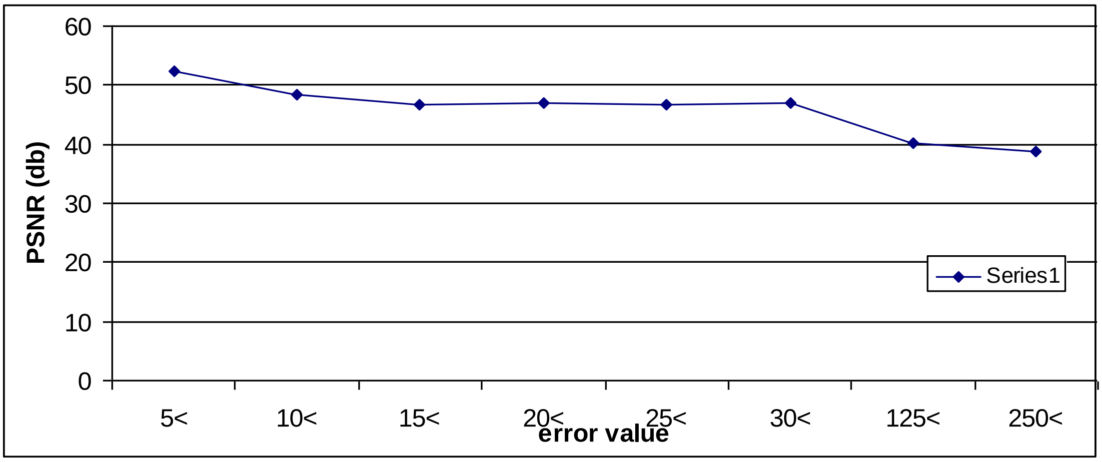


Fig. 6 The relationship between PSNR and error value.

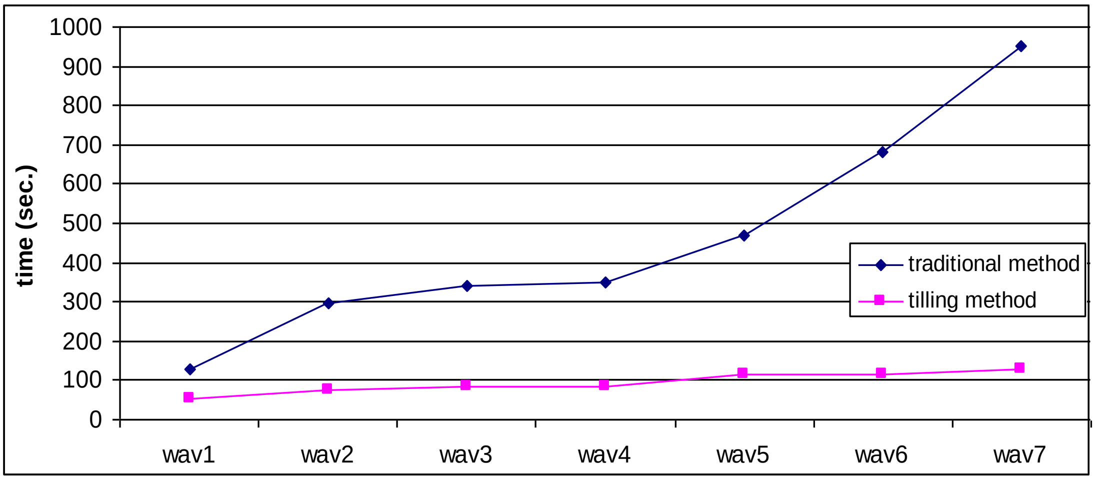


Fig. 7 A Comparing between the obtained coding time for traditional and tilling

TABLE I

SHOWS THE TIME, MSE AND PSNR FOR CONSTANT BLOCK SIZE AND VARIANT DATA FILE.

FILE NAME	DATA SIZE	MSE	PSNR	TIME
wav1	61.7	2.7745	43.69896	129.2009
wav2	92.9	2.0505	45.01221	298.5246
wav3	99.7	4.9336	41.19916	340.8337
wav4	101	11.0329	37.70391	350.4971
Wav5	117	3.5458	42.63366	467.6696
Wav6	140	25.3756	34.08664	682.3705
Wav7	166	4.7006	41.40927	951.6002

TABLE II
SHOWS THE TIME, MSE AND PSNR FOR CONSTANT BLOCK SIZE AND VARIANT ERROR VALUE (X^2).

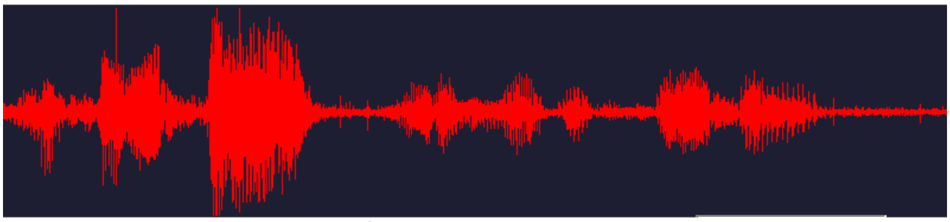
CHI VALUE	MSE	PSNR	TIME
5<	0.3714	52.43238	373.9372
10<	0.942	48.39029	279.8777
15<	1.3888	46.70441	211.6549
20<	1.2658	47.10715	160.9309
25<	1.3752	46.74714	126.3787
30<	1.3414	46.85522	100.7952
125<	6.3669	40.09152	26.1063
250<	8.4934	38.83999	5.4903

TABLE III
SHOWS THE TIME FOR TRADITIONAL AND TILLING METHOD WITH CONSTANT BLOCK SIZE.

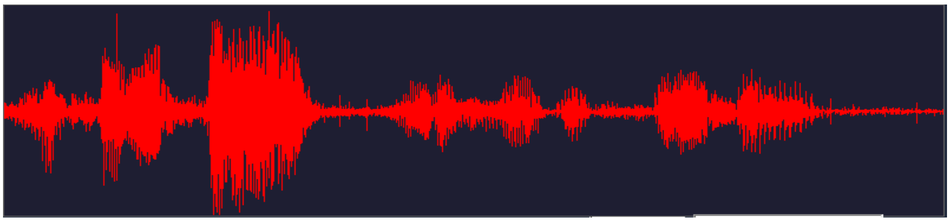
FILE NAME	TRADITIONAL TIME	TILLING TIME	PROPORTIONAL PERCENTAGE
wav1	129.2009	51.062	71.67359
wav2	298.5246	75.991	79.70952
wav3	340.8337	85.0357	80.03245
wav4	350.4971	83.591	80.74331
wav5	467.6696	115.2056	80.23495
wav6	682.3705	115.3845	85.53635
wav7	951.6002	129.06	88.0573

TABLE IV
SHOWS THE TIME, MSE AND PSNR FOR VARIANT BLOCK SIZE.

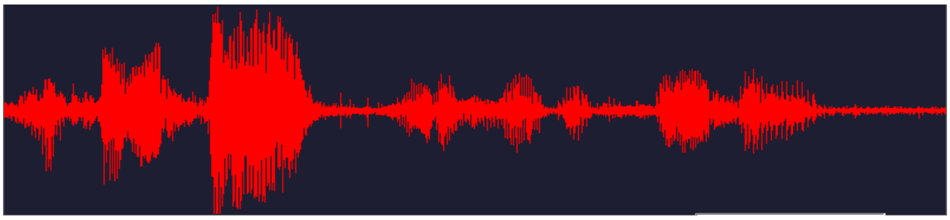
FILE NAME	BLOCK SIZE	MSE	PSNR	TIME
wav1	12	1.616	46.04639	140.4321
wav2	12	2.0065	45.10641	319.2361
wav3	12	3.157	43.13806	367.37
wav4	12	7.9344	39.13566	379.224
wav5	12	3.2587	43.00036	508.6944
wav6	12	16.9268	35.84505	734.5547
wav7	12	1.9097	45.32115	1028.948
wav1	16	2.7745	43.69896	129.2009
wav2	16	2.0505	45.01221	298.5246
wav3	16	4.9336	41.19916	340.8337
wav4	16	11.0329	37.70391	350.4971
wav5	16	3.5458	42.63366	467.6696
wav6	16	25.3756	34.08664	682.3705
wav7	16	4.7006	41.40927	951.6002
wav1	20	3.1645	43.12775	123.4149
wav2	20	4.5054	41.59347	282.651
wav3	20	11.0363	37.70257	324.0746
wav4	20	15.9848	36.09373	333.1759
wav5	20	13.8695	36.7102	444.6526
wav6	20	36.9334	32.45661	647.3813
wav7	20	12.8649	37.03674	897.5963
wav1	32	21.5701	34.79228	114.8813
wav2	32	4.7543	41.35994	257.7711
wav3	32	11.719	37.4419	302.0199
wav4	32	16.2418	36.02446	307.7313
wav5	32	20.5915	34.99392	416.0394
wav6	32	42.1466	31.88318	589.5038
wav7	32	20.4715	35.01931	837.7698



a



b



c

Fig.10 The waveform of wav1 (a. orignal, b. obtained from traditional method, c. obtained from tiling).

VI. CONCLUSION

The suggest improved Fractal coding method gives good reduction in encoding time. The obtained results for different tests applied on different wave files stimulated a number of conclusions which classified into two classes as the following.

1. The effects of block size and max error value on the MSE and PSNR are:
   a) The PSNR value increases where as block size value decrees.
   b) The PSNR value increases where as max error value decrees.
2. The time enhancement obtained from applying the enhanced PIFS method, is summarized by
   a) The encoding time depends on the data size.
   b) The encoding time increase on when the block size is decreased.
   c) The encoding time increases with the decreased or max. Error decreased.

Finally, for the adaptive method the time of encoding reduced from 70%-80% of the time consumed of traditional coding.

REFERENCES

[1] L.R. Rabiner, R.W.Schafer, *Introduction to Digital Speech Processing*, Foundations and Trends in Signal Processing, Vol.1, Nos, 1-2 (2007).

[2] E. A. Al-Hilo, L. E. George, *Speeding-up Fractal Colored Image Compression Using Moments Features*, DOI 10.1109/DICTA.2008.18,IEEE.

[3] M. Al-Akaidi, *Fractal Speech Processing*, ISBN-I3 978-0-5II-I9439-9, May, 2004.

[4] H. Xiao, *Fractal Audio Coding*, M. Sc., Queen's University Kingston, July 2005.

[5] L.E. George, *IFS Coding for Zero-mean Image Blocks*, Iraqi Journal of Science, Vol. 47, No.1. 2006, pp190-194.

[6] S.C. Pei, C.C. Tseng, C.Y. Lin, *A Parallel Decoding Algorithm for IFS Codes Without Transient Behaviour*, IEEE Transaction on Image Processing, Vol.5, No.3,March 1996.

[7] W. F. Jassim, *Fractal Audio Compression*, M. Sc., thesis, college of Science, Nahrain

