RESEARCH ARTICLE OPEN ACCESS

Dynamic Texture Coding using Modified Haar Wavelet with CUDA

Premanand P Ghadekar*, Nilesh A Parmar**, Nilkanth B Chopade***
*(Department of Applied Electronics, SGBA University, Amravati-02)
** (Department of Electronics Engineering, VIT college, Pune-37)
***(Department of Electronics Engineering, PCCOE college, Nigadi, Pune-44)

ABSTRACT
Texture is an image having repetition of patterns. There are two types, static and dynamic texture. Static texture is an image having repetitions of patterns in the spatial domain. Dynamic texture is number of frames having repetitions in spatial and temporal domain. This paper introduces a novel method for dynamic texture coding to achieve higher compression ratio of dynamic texture using 2D-modified Haar wavelet transform. The dynamic texture video contains high redundant parts in spatial and temporal domain. Redundant parts can be removed to achieve high compression ratios with better visual quality. The modified Haar wavelet is used to exploit spatial and temporal correlations amongst the pixels. The YC_bC_r color model is used to exploit chromatic components as HVS is less sensitive to chrominance. To decrease the time complexity of algorithm parallel programming is done using CUDA (Compute Unified Device Architecture). GPU contains the number of cores as compared to CPU, which is utilized to reduce the time complexity of algorithms.

Keywords **–** Compression, CUDA, Dynamic texture, GPU, Haar wavelet, PSNR, SSIM

I. INTRODUCTION

Video compression is a fast growing field with many video compression algorithms are available. The video contains highly correlated neighboring pixels which are redundant. This redundancy can be removed to achieve higher compression ratio [1].

The dynamic texture contains more redundant part as compared to video. This redundant part can be reduced. The dynamic texture gives high compression ratio as compared to video. The visual quality is also good as compared to video.

In the proposed algorithm, the 2-D modified Haar wavelet is used. It is a simple and efficient averaging differencing method applied in rows and columns to achieve high compression ratios with good visual quality.

GPU [2] is also called visual processing unit (VPU). Originally GPUs were manufactured for appropriate rendering of gaming graphics, but now they are applied to scientific calculations also. GPU has an extremely parallel array of integer and floating-point processors with dedicated high-speed memory. GPU has a parallel architecture that executes many concurrent threads slowly instead of executing a single thread very quickly. As shown in fig.1, GPU is a co-processor to the CPU. Data and functions should be off-loaded to the GPU from CPU. Kernel is a function (code) written for execution on the GPU, and it is executed as many different threads in parallel. Core is a single independent computational unit within a CPU or GPU chip. CPU core performs general operations whereas GPU core performs specialized operations. CPU has four cores, and GPU has 240 to 2048 cores. A GPU accelerates an application that satisfies the following criteria: 1) The computations divided into a large number of independent units of work. 2) The time consumed in computation significantly surpasses the time consumed in transferring data to and from GPU memory. Programming is done in CUDA [3] to reduce the time complexity of the proposed algorithm. CUDA allows parallel programming in CPU and GPU.

II. LITERATURE SURVEY

JPEG uses the Discrete Cosine transform which gives a high compression ratio but have disadvantages like blocking artefacts, graveness and blurred video [4]. Traditional compression algorithms have high time complexity. Traditional algorithms only exploit spatial and temporal correlation among pixels but chromatic correlation among pixels are not used.

Scalable Accordion-DCT based video coding method [5] converts 3D video into 2D structure. It allows extracting the temporal redundancy of video using up/down sampling method (SVC) which is based on a combination of the forward and backward DCT. Such technique prevents the motion compensation step and decreases the number of computations required. This method transforms multiple frames into one large frame, and scalable Accordion-DCT is applied on it. In Accordion-DWT [6], 2D Daubechies wavelet transform applied to

large frame instead of up/down sampling method. It overcomes the limitations of Accordion-DCT by getting high compression ratio without much affecting the number of computations.

The YC_bC_r color model is used to exploit chromatic correlation among pixels; C_b and C_r components down-sampled by half without loss of visual quality as a change in chromaticity are not visible to human eyes. 2-D Haar wavelet is simple and efficient transform to achieve high compression ratios without loss of visual quality. 2-D Haar wavelet is computationally less complex as compared to other transform as only averaging differencing is used, and no temporary variable is used during transform [7]. Traditional algorithms have a high time complexity which reduced by heterogeneous programming (CPU+GPU) in CUDA [8].

III. Haar wavelet

Wavelet indicates "let the wave go" because wavelet considers finite range of signal instead of its infinite length. Fourier transform splits the signal into a sequence of sine waves of different frequencies, whereas the wavelet transform splits the signal into its "wavelets", scaled and shifted versions of the mother wavelet. The wavelet is irregular in shape and compactly supported. These properties make wavelets an ideal tool for analysing signals of a non-stationary nature. Their irregular shape enables them to analyse signals with discontinuity or sharp changes, while their compactly supported nature provides temporal localisation of a signal's features. The wavelet transform provides localisation of a signal in both time and frequency domain which is very important to evaluate signals of the non-stationary nature. The location of the wavelet allows explicitly to represent the location of events in time. The shape of the wavelet allows representing different detail or resolution.

Alfréd Haar proposed the Haar wavelet in 1910. It is irregular and not continuous. The Haar basis is obtained with a multi-resolution of piecewise constant functions. The scaling function is $\Phi=1_{[0,\ 1]}$. The filter h[n] = has value $2^{-1/2}$ $< 2^{-\frac{1}{2}} \emptyset\left(\frac{t}{2}\right), \emptyset(t-n) >$ at n=0 and n=1. Hence,

$$\frac{1}{\sqrt{2}}\psi\left(\frac{t}{2}\right) = \sum_{n=-\infty}^{+\infty} (-1)^{1-n} h[1-n]\phi(t-n) = \frac{1}{\sqrt{2}}\left(\phi(t-1) - \phi(t)\right) \quad (1)$$

So,

$$\psi = \begin{cases} -1 & if\ 0 \le t < 1/2 \\ 1 & if \frac{1}{2} \le t < 1 \\ 0 & otherwise \end{cases}$$

The Haar wavelet is the only real compactly supported wavelet that is symmetric or anti-symmetric and which generates an orthogonal basis of Hilbert space L2(R) [9]. Haar wavelet is simple averaging and differentiating method which is applied on either rows or column. The 2D-modified Haar wavelet is same as Haar wavelet difference is that it is applied on rows and then on the column. As only averaging and differentiating is used the image quality does not decrease [10].

The 2D-modified Haar wavelet [11] has properties like it is real and orthogonal. The basis vectors of the Haar transform are sequentially ordered. The original signal is split into low and high-frequency part. It has linear phase, compact support, the and perfect representation properties.

IV. Encoder

Step 1: Take input as uncompressed video (.avi).
Step 2: Frames are extracted from Video. Each Frame is converted from RGB to YC_bC_r.
Step 3: The C_b and C_r components are down-sampled by factor 2. Down-sample by factor 2 means only considering alternate row and column of the image matrix. Down sampling C_b and C_r components does not affect the video quality because human eyes are less sensitive to Chrominance and its effect is negligible to the human eye. Suppose C_b and C_r components have 512*512 pixel matrix then after down sampling it becomes 256*256.
Step 4: Each Image component is divided into Bock size of 8 by 8.Instead of applying DWT on the whole image we use it on 8*8 blocks because neighboring pixels are highly correlated.
Step 5: Algorithm to find out block redundancy is as follows:

1. Apply DWT to 8*8 blocks.
2. Calculate summation of pixels in LL band and store in variable say sum.
3. Apply DWT to other blocks, and store summation of LL band pixels invariable say sum1
4. If sum = sum1 +/- error, error=1,2,3,4,5 then block is redundant
5. Else not redundant

If a block is redundant store only references of that block i.e. frame no and block no. else perform following steps.
Step 6: Algorithm for 2D-modified Haar wavelet

1. Find the mean of each pair of samples.
2. Find the difference between each average.
3. Replace the first half of the array by average values.
4. Replace the second half of the array by difference values.
5. Repeat the procedure on the first half of the array.

Step7: Quantization is used to achieve lossy compression, the algorithm is as follows:

1. Choose Delta value between 1 to 16
2. All pixels having values between minus Delta to plus Delta are randomized to zero.

As the value of Delta increases compression ratio increases and video quality decrease. Delta = 4 gives optimal results.
Step 8: Instead of applying RLC on rows or column, applying on zigzag scanned matrix provides more zero count. The advantage of zigzag scan is that we access the neighboring pixels which are highly correlated.
Step 9: It is applied to Zigzag scanned matrix. RLC gives output in the form of pairs (Pixel value, count).Here count means the no of times pixel value repeated.
Step 10: The run length pairs stored in compressed .txt File.

V. Decoder

Step 1: If the reference found in the compressed file, then copy Run length values of that block.
Step 2: (Pixel Value, Count) converted into pixel value and pixel value is copied count no of times.
Step 3: The block traversed in reverse zig zag style.
Step 4: Reverse algorithm of DWT is applied.
Step 5: Up sampling the C_b and C_r components by factor 2.Take first row as output, take an average of first and second row as output and take a second row as output. Perform this for all rows and column. Suppose C_b and C_r components have 512*512 pixel matrix then after up sampling it becomes 1024*1024.
Step 6: YC_bC_r frames are converted to RGB frames.
Step 7: All frames are collected to get reconstructed video.

VI. Parallel Programming

To achieve better performance implementation is done in CUDA architecture. The use of GPUs for computations and calculations provides a platform to write applications that can run on a large number of cores. CUDA is a NVIDIA's employment of parallel architecture [12] on their GPUs and offers APIs for programmers to develop parallel applications using the C programming language with some constraints and CUDA extensions. CUDA delivers a software abstraction for the hardware called grid. Grid is a group of blocks which are a group of threads that can share memory as depicted in fig.2. These threads are then assigned to multiple cores (processors). Eight scalar processors constitute a multiprocessor, and different models of GPUs contain various counts of multiprocessor.

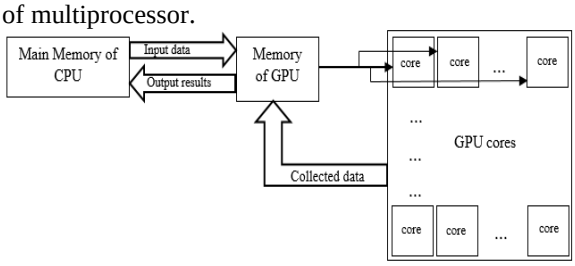

Fig.1: CPU and GPU interfacing

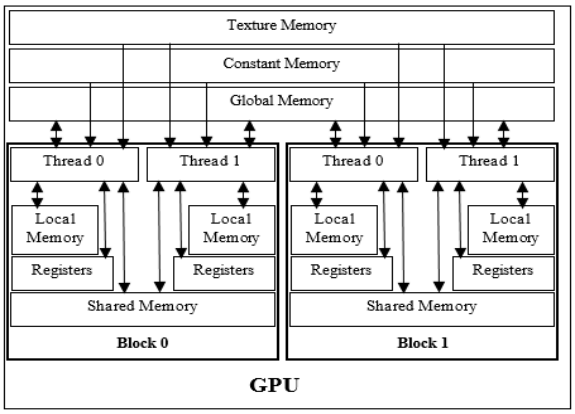

Fig.2: Different memories of GPU

VII. Results

The proposed algorithm is implemented using CUDA on different videos and dynamic textures. CPU used for processing is 64-bit Intel Core i3 with 2.10GHz speed and 3 GB RAM. To parallelize code GPU version Geforce GTX 525M with 96 processing cores and 1 GB dedicated graphics memory is used. Dynamic textures with different frame size are taken from Renaud Péteri et al.'s dynamic texture database [13]. Parameters like PSNR and SSIM are considered to compare visual quality of original and reconstructed videos. Compression Ratio (CR) represents the efficiency of proposed algorithm with considerable visual quality. Table-1 and table-2 display CR, PSNR and SSIM for different videos without and with YC_bC_r respectively. Fig.3 and fig.4 exhibit graphs of original and compressed size for different videos without and with YC_bC_r respectively. Table-3 and table-4 represent CR, PSNR and SSIM for standard dynamic textures without and with YC_bC_r respectively. Fig.5 and fig.6 exhibit graphs of original and compressed size for different textures without and with YC_bC_r respectively.

Table 1: Compression ratio, PSNR and SSIM for different RGB videos

Video	Size (MB)	Compressed Size (MB)	Compression Ratio	PSNR (dB)	SSIM
United (720*420)	838	115.10	86%	35.34	0.97
Bbb (624*272)	773	157.75	79.59%	20.92	0.86
Hf (608*240)	772	78.85	89.78%	21.48	0.86
Viptrain (360*240)	154	49.05	68%	34.42	0.93
Sample (512*512)	320	74.24	76.8%	28.12	0.85

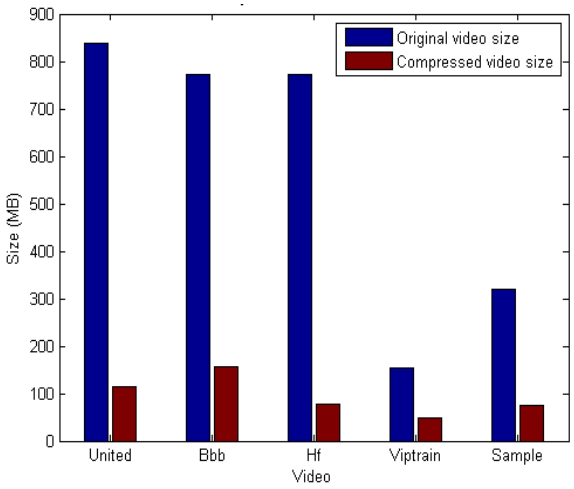

Fig. 3: Original and compressed size of different videos

Table 2: Compression ratio, PSNR and SSIM for different RGB videos with YCbCr

Video	Size (MB)	Compressed Size (MB)	Compression Ratio	PSNR (dB)	SSIM
United (720*420)	838	74.88	91.06%	33.34	0.97
Bbb (624*272)	773	104.03	86.54%	19.92	0.86
Hf (608*240)	772	78.53	89.82%	20.38	0.86
Viptrain (360*240)	154	32.35	78.99%	32.12	0.88
Sample (512*512)	320	48.92	84.71%	28.12	0.85

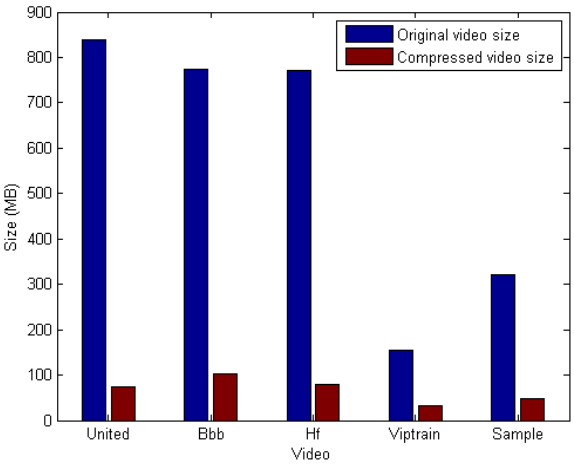

Fig. 4: Original and compressed size of different videos with YC_bC_r

Table 3: Compression ratio, PSNR and SSIM for different dynamic textures

Video	Size (MB)	Compressed Size (MB)	Compression Ratio	PSNR (dB)	SSIM
Steps (355*288)	29.6	3.56	87.97%	42.77	0.97
Clouds (320*288)	55.1	5.48	90.05%	36.99	0.971
Camel (320*288)	55.1	5.15	90.65%	35.27	0.96
Bgrass (360*240)	72.8	8.78	87.93%	37.67	0.965
Bird (320*240)	23.2	4.96	78.62%	39.75	0.988

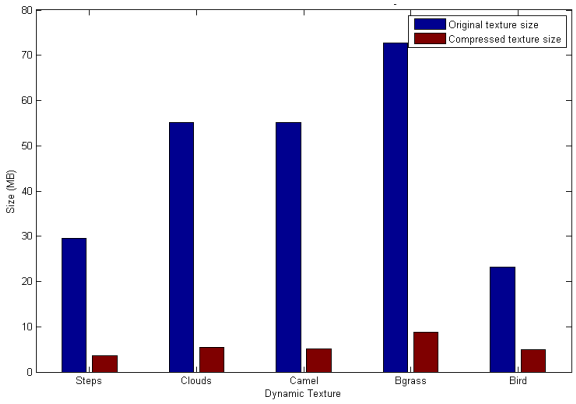

Fig. 5: Original and compressed size of different dynamic textures

Table 4: Compression ratio, PSNR and SSIM for different dynamic textures with YCbCr

Video	Size (MB)	Compressed Size (MB)	Compression Ratio	PSNR (dB)	SSIM
Steps (355*288)	29.6	2.34	92.04%	41.57	0.96
Clouds (320*288)	55.1	3.62	93.43%	35.99	0.97
Camel (320*288)	55.1	3.40	93.82%	34.07	0.95
Bgrass (360*240)	72.8	5.79	92.04%	37.03	0.95
Bird (320*240)	23.2	3.27	85.90%	38.05	0.97

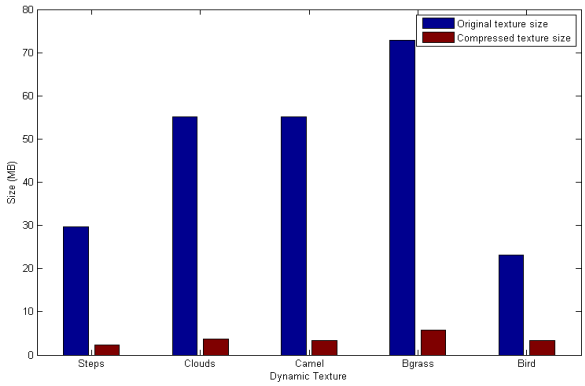

Fig. 6: Original and compressed size of different dynamic textures with YC_bC_r

Table-5 shows comparison of time taken by CPU only and CPU with GPU for different dynamic textures. Fig. 7 demonstrates the same using graph.

Table 5: CPU and CPU+GPU time for different dynamic textures

Dynamic Texture	CPU time(Sec)	CPU+GPU time (Sec)	%ET reduced
Bgrass	170.37	151.14	11.3
Chair	184.38	159.20	13.7
DNA	159.22	143.08	10.1
Grass_water	39.96	35.36	11.5
Duck_bush	133.58	118.93	10.9

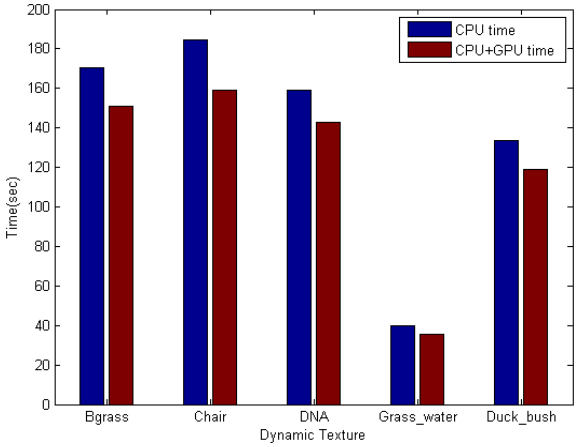

Fig. 7: Comparison of CPU and CPU+GPU time

VIII. CONCLUSION

The 2D-modified Haar wavelet gives high compression ratio with good PSNR ratio. The 2D-modified Haar wavelet is simple and efficient transform to implement which only have averaging and differing, and no temporary array required for transform. The proposed algorithm increases compression ratios by using different quantization levels.

The texture video contains more no of redundant blocks as compared to video. Therefore, higher compression ratio can be achieved in the range 85%-90%. The PSNR values for texture video are high which shows that video quality of the reconstructed video is good.

The proposed algorithm reduces the time complexity. CUDA allows parallel programming in CPU and GPU. GPU contains large number of cores as compared to CPU allowing multiple blocks to run parallel.

REFERENCES

[1] Zhang Mark Nelson, "*The Data Compression Book: Featuring Fast, Efficient Data Compression Techniques in C*", M&T Books, 1992, ISBN 97815585 12160.

[2] Nvidia GPU Programming Guide GeForce 8 and 9 Series. (developer. download. nvidia.com)

[3] NVIDIA CUDA Programming Guide, Version 2.1. (docs.nvidia.com)

[4] K. R. Rao and P. Yip, Discrete Cosine Transform: Algorithms, Advantages, Applications.Boston: Academic Press, 1990.

[5] G.Suresh, P.Epsiba, Dr.M.Rajaram, Dr.S.N. Sivanandam, "*Scalable ACC-DCT Based Video Compression Using Up/Down Sampling Method*", International Journal of Computer and Network Security, Vol. 2, No. 6, June 2010

[6] N R Gawande, P P Ghadekar and M L Dhore,"*Efficient Accordion DWT based Video Coding*", International Journal of Computer Applications 100(10):33-39, August 2014.

[7] Talukder, Kamrul Hasan, and Koichi Harada. "*Haar wavelet based approach for image compression and quality assessment of compressed image."* arXiv preprint arXiv: 1010.4084 (2010).

[8] NVIDIA CUDA C Programming Best Practice Guide. (docs.nvidia.com)

[9] Stephane Mallat, "*A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way 3rd*", Academic Press ©2008, ISBN 9780123743701.

[10] Stanković, Radomir S., and Bogdan J. Falkowski. "*The Haar wavelet transform: its status and achievements*." Computers & Electrical Engineering 29.1 (2003): 25-44.

[11] P Raviraj and M Y Sanavullah, "*The Modified 2D-Haar WAVELET Transformation in Image Compression*", Middle East Journal Scientific Research 2(2),73-78, 2007.

[12] Michael J. Quinn, "*Parallel computing (2nd ed.): theory and practice*", McGraw-Hill, Inc. New York, USA ©1994, ISBN: 0-07-051294-9.

[13] Péteri, Renaud, Sándor Fazekas, and Mark J. Huiskes. "*DynTex: A comprehensive database of Dynamic Textures*." Pattern Recognition Letters 31.12 (2010): 1627-1632.