Effective Compression of Digital Video using LZW

[1]Komal Khedekar, [2]Amruta Kadu, [3]Namrata Raut,
[4]Suraj Deshmukh, [5]Aniruddha Kale, [6]Anil Lohar
[1,2,3,4,5]Student, [6] Assistant Professor
[1,2,3,4,5,6]Computer Engineering,
[1]ABMSP's Anantrao Pawar College of Engineering & Research, Pune, India

Abstract : Video compression is nothing but compression of video, it involves compression of video size, audio format. In other words we can state video compression as one of the encoding format of video that it can have less memory size than the original video file. One of the basic need for video compression is nothing but, if video size less is then it is easy to transmit over the air i.e. on internet. Video compression is done by removing repetitive frames and audio effects so, mainly it will reduce all unimportant data or it will reduce it from video file. There are different algorithms for compression of video. We are going to use LZW compression algorithm to do effective compression of digital video. LZW is dictionary based algorithm. This paper implements one of the advanced video compression technique by providing less data loss by using LZW algorithm.

IndexTerms **- Image Processing, Video Processing, Video Compression, Video Quality, Decompression**.

I. INTRODUCTION

As we know the basic meaning of video compression it is nothing but reducing the size of video. The video consist of multiple number of frames, which are aligned one after another and move at a certain speed, that speed can be 30fps,25 fps(fps is frames per second). There are two categories of compression techniques, lossy and lossless. We can use different techniques to compress files, both have the same aim: To look for duplicate data in the graphic (GIF for LZW) and use a much more compact data representation. Lossless compression reduces bits by identifying and eliminating statistical redundancy. No information is lost in lossless compression. On the other hand, Lossy compression reduces bits by removing unnecessary or less important information. So we need Data Compression mainly because: Uncompressed data can take up a lot of space, which is not good for limited hard drive space and internet download speeds. While hardware gets better and cheaper, algorithms to reduce data size also helps technology evolve. Example: One minute of uncompressed HD video can be over 1 GB. How can we fit a two-hour film on a 25 GB Blu-ray disc? Lossy compression methods include DCT (Discreet Cosine Transform), Vector Quantization and Huffman coding while Lossless compression methods include RLE (Run Length Encoding), string-table compression, LZW (Lempel Ziff Welch) and zlib. There Exist several compression Algorithms, but we are concentrating on LZW.

LZW algorithm is created by Abraham Lempel, ziv and terry Welch. This implements data compression, which is difficult task to perform. This is dictionary based algorithm. This algorithm is mainly used for GIF and TIFF image format. In this the frame is captured and scanned for repetitive sequence of data into that frame. Then these sequence of data from that frame are stored in dictionary and then the algorithm is performed. There are some of the advantages and disadvantages of this LZW, the video size is reduced without loss of data by reducing repetitive data and monochrome data, this algorithm performs faster and easy to apply. It provides decompression technique also LZW is useful as it do not need to pass the string table to decompression code. Table can be recreated using input stream. It avoids insertion of large input streams. Algorithm works good for text files than the image or video files.

II. LEMPEL–ZIV–WELCH (LZW) ALGORITHM

The LZW algorithm is a very common compression technique. This algorithm is typically used in GIF and optionally in PDF and TIFF. Unix's 'compress' command, among other uses. It is lossless, that means data is lost when compressing. The algorithm is simple to implement and has the potential for very high throughput in hardware implementations. This algorithm is widely used in Unix file compression utility, and is used in the GIF image format. The Idea relies on reoccurring patterns to save data space. LZW is the foremost technique for general purpose data compression due to its simplicity and versatility. It is the basis of many PC utilities that claim to "double the capacity of your hard drive".

- Advantages of LZW over Huffman:

1. LZW requires no prior information about the input data stream.
2. LZW can compress the input stream in one single pass.
3. Another advantage of LZW its simplicity, allowing fast execution.

The idea of the compression algorithm is the following: as the input data is being processed, a dictionary keeps a correspondence between the longest encountered words and a list of code values. The words are replaced by their corresponding codes and so the input file is compressed. Therefore, the efficiency of the algorithm increases as the number of long, repetitive words in the input data increases. LZW compression works by reading a sequence of symbols, grouping the symbols into strings, and converting the strings into codes. Because the codes take up less space than the strings they replace, we get compression.

LZW compression uses a code table, with 4096 as a common choice for the number of table entries. Codes 0-255 in the code table are always assigned to represent single bytes from the input file.

A. Encoding:

When encoding begins the code table contains only the first 256 entries, with the remainder of the table being blanks. Compression is achieved by using codes 256 through 4095 to represent sequences of bytes.

- Encoding Algorithm:

1. Initialize table with single character string.
2. Take "P" as the first input character.
3. Check whether table is empty or not.
4. If table is not empty, then take "C" as next input character.
5. If P+C exists in the table, then P = P + C.
6. Else it will return the codeword for P.
7. Add P + C to the string table.
8. If P= C, then end of the string.
9. Return code for whole input string.

- Compression using LZW:

Example 1: Use the LZW algorithm to compress the string: BABAABAAA
The steps involved are systematically shown in the diagram below:

Step 1: BABAABAAA P=A , C=empty

Encoder	Output	String	Table
Output code	Representing	Codeword	String
66	B	256	BA

Step 2: BABAABAAA P=B , C=empty

Encoder	Output	String	Table
Output code	Representing	Codeword	String
66	B	256	BA
65	A	257	AB

Step 3 : BABAABAAA P=A , C=empty

Encoder	Output	String	Table
Output code	Representing	Codeword	String
66	B	256	BA
65	A	257	AB
256	BA	258	BA

Step 4: BABAABAAA P=A , C=empty

Encoder	Output	String	Table
Output code	Representing	Codeword	String
66	B	256	BA
65	A	257	AB
256	BA	258	BA
257	AB	259	ABA

Step 5: BABAABAAA P=A , C=C

Encoder	Output	String	Table
Output code	Representing	Codeword	String
66	B	256	BA
65	A	257	AB
256	BA	258	BA
257	AB	259	ABA
65	A	260	AA

Step 6: BABAABAAA P=AA , C=empty

Encoder	Output	String	Table
Output code	Representing	Codeword	String
66	B	256	BA
65	A	257	AB
256	BA	258	BA
257	AB	259	ABA
65	A	260	AA
260	AA		

B. Decoding

Decoding is achieved by taking each code from the compressed file and translating it through the code table to find what character or characters it represents. The LZW decompressor creates the same string table during decompression. It starts with the first 256 table entries initialized to single characters. The string table is updated for each character in the input stream, except the first one. Decoding achieved by reading codes and translating them through the code table being built.

- Decoding Algorithm:

1. Initialize table with single character string.
2. Take "OLD" as the first input code.
3. Convert the codeword to string character.
4. If the input string is not empty, then take "NEW" as new character.
5. If "NEW" does not exist in the string, then S = S + C.
6. Else it will return the codeword for S.
7. Take "C" first character of S.
8. Add "OLD + C" to the string table
9. If OLD = NEW, then end of the string.

- Decompression using LZW:

Use LZW to decompress the output sequence of : <66><65><256><257><65><260>
The steps involved are systematically shown in the diagram below.

LZW compression step 1

<66><65><256><257><65><260> Old=65 S=A New=66 C=A

Encoder Output	String	Table
String	Codeword	String
B		
A	256	BA

LZW compression step 2

<66><65><256><257><65><260> Old=256 S=BA New=256 C=B

Encoder Output	String	Table
String	Codeword	String
B		
A	256	BA
BA	257	AB

LZW compression step 3

<66><65><256><257><65><260> Old=257S=AB New=257C=A

Encoder Output	String	Table
String	Codeword	String
B		
A	256	BA
BA	257	AB
AB	258	BAA

LZW compression step 4

<66><65><256><257><65><260> Old=65 S=A New=66 C=A

Encoder Output	String	Table
String	Codeword	String
B		
A	256	BA
BA	257	AB
AB	258	BAA
A	259	ABA

LZW compression step 5

<66><65><256><257><65><260> Old=260 S=AA New=260 C=A

Encoder Output	String	Table
String	Codeword	String
B		
A	256	BA
BA	257	AB
AB	258	BAA
A	259	ABA
AA	260	AA

III. PROPOSED SYSTEM

In the proposed system DCT (Discrete cosine transform) is used to extract the frames and temporal & spatial redundancy per frame , and LZW (Lempel Ziv Welch) algorithm is used for actual compression of the video. In DCT , first

extract the frames then convert the RGB into the YUV color. After color conversion spatial redundancy is done within the frame and temporal redundancy between the frames. Then apply the LZW algorithm on each frame and reduced the data as per their redundancy. Here no data loss hence it can be provide the better quality as carry the original data after compression also. This proposed system provide the compressed video with good compression ratio and provides the similar quality as original video.

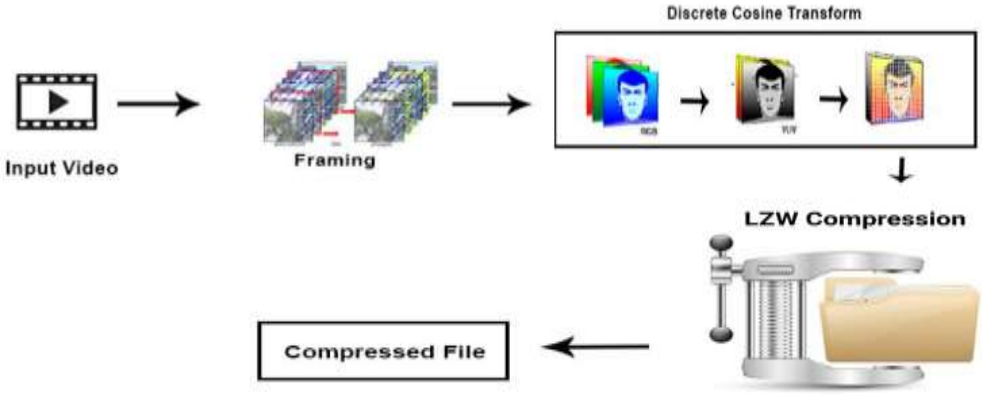

Fig. Architecture diagram.

- Output of implemented System:

Original video Size	Compressed video Size
15.4 MB	3.78 MB
12.9 MB	5.16 MB
37.4 MB	23.4 MB

Table 1: Output of implemented system

IV. EXPERIMENTAL WORK

To measure the quality of our system to another apps, same input video given to the different apps as well as to proposed system. Result as shown in below. And Proposed system gives the better result than other apps with respect quality as well as compression ratio also.

- Comparison of Video Compression apps:

Available Apps	Video 1(15.4MB)	Video 2(12.9MB)	Compression Ratio
Super Video Compressor app	6.48 MB	5.89 MB	42.07%
Video Compressor app	4.09 MB	6.07 MB	26.55%
Video Editor app	7.97 MB	10.0 MB	51.75%
Video Smaller online	4.47 MB	6.25 MB	29.02%
Implemented System Output	**3.78 MB**	**5.16MB**	**24.54%**

To measure the quality of the compressed video with respect to original video *VQMT* (Video Quality Measurement Tool) is used. This software provides fast implementations of the following objective metrics:

- PSNR: Peak Signal-to-Noise Ratio,

- SSIM: Structural Similarity,
- MS-SSIM: Multi-Scale Structural Similarity,
- VIFp: Visual Information Fidelity, pixel domain version,
- PSNR-HVS: Peak Signal-to-Noise Ratio taking into account Contrast Sensitivity Function (CSF),
- PSNR-HVS-M: Peak Signal-to-Noise Ratio taking into account Contrast Sensitivity Function (CSF) and between-coefficient contrast masking of DCT basis functions.

In this software, the above metrics are implemented in OpenCV (C++) based on the original Matlab implementations provided by their developers. The source code of this software can be compiled on any platform and only requires the OpenCV library (core and imgproc modules). This software allows performing video quality assessment without using Matlab and shows better performance than Matlab in terms of run time.

Fig. comparison of compressed video with original video

V. CONCLUSION

In this paper , proposed system consist of effective video compression. LZW algorithm is used to effective compression of digital video which provides the similar quality as the original video. LZW algorithm is suitable for where the data redundancy is present. In the video frames are repeated with the minor difference, hence it is effective for the video compression. It is lossless algorithm ,only redundancy us eliminated not data.

REFERENCES

[1] U.Thavamani , Dr.K.Mahesh "*A Robust Video Compression Method Based On LZW Encoding in 2D-DCT Domain*" , International Journal of Advanced Research Trends in Engineering and Technology , Vol. 3, Special Issue 20, April 2016

[2] Ms. S. S. Wadd, and Prof. Mrs. S. B. Patil, "*Video Compression using DCT*", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 9, ISSN: 2277 128X, September 2014.

[3] Saumya Mishra , Mr. Avinash Singh , " Image compression and enhancement by using the LZW and BHEPL" International Journal of Scientific and Research Publications, Volume 7, Issue 5, May 2017