www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 8 Issue 3 March 2019, Page No. 24575-24585
Index Copernicus Value (2015): 58.10, 76.25 (2016) DOI: 10.18535/ijecs/v8i03.4296

Framework for Lossless Data Compression Using Python

Manas Malik

Student
Department of Computer Science and Engineering
School of Engineering and Technology, Jain University

Abstract:
A lot has been done in the field of data compression, yet we don't have a proper application for compressing daily usage files. There are appropriate and very specific tools online that provide files to be compressed and saved, but the content we use for streaming our videos, be it a Netflix video or a gaming theater play, data consumed is beyond the calculation of a user. Back-end developers know all about it and as developers we have acknowledged it but not yet achieved it in providing on an ease level. Since the user would not never be concerned about compression, developers can always take initiative while building the application to provide accessibility with compression before-hand. We have decided to create a framework that will provide all the functionality needed for a developer to add this feature. Making use of the python language this process can work. I'm a big fan of Python, mostly because it has a vibrant developer community that has helped produce an unparalleled collection of libraries that enable one to add features to applications quickly. For the DEFLATE lossless compression, has a higher level of abstraction provided by the zlib C library, in Python it is generally provided by the Python zlib library which is an interface, we have a lot to do including the audio, video and subtitles of the file. We also make use of the fabulous ffmpy library. ffmpy is a Python library that provides access to the ffmpeg command line utility. ffmpeg is a command-line application that can perform several different kinds of transformations on video files, including video compression, which is the most commonly requested feature of ffmpeg. Frame rate and audio synchronization are few other parameters to look closely. This is an ongoing project and there remains few implementation aspects, data compression remains a concern when touched upon the design. We along with python community intend to solve this issue.

1. Introduction

This paper includes the study of different compression techniques that are currently in use. Making use of all the existing libraries and theoretical concepts, we intent to create a new framework for developers, this will be the first in Python language that would provide all the necessary features for compression. Most importantly video files that consume data of use on daily basis. Handling each frame of the video is a very efficient and lossless process, so far it does not has any fast providing application. The field additionally demands collaboration of other tools outside compression tools existing. As Python is becoming more handy to developers, it is very convenient for the developers to grasp algorithms and develop in the language. All the theoretical concepts are considered based on the study emerged so far, practical libraries used are very specific and not quite efficient.

2. Compression Techniques

Number of pixels can be reduced using video compression, we can achieve this by reducing all the irrelevant and unnecessary information. Compression technique purposes include:

- Colour resolution can be reduced along with the noise or the unnecessary colour information.
- To have no invisible portion in the image.

- In comparing the adjacent frames remove the details that remain unchanged.

The algorithm and techniques used for the video compression are depicted in the Fig.1.

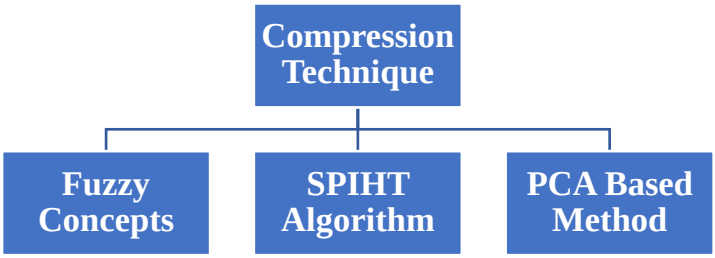


Fig1: Compression techniques categories

2.1 Fuzzy Concepts

For providing an optimal and efficient compression, grey image compression techniques are proposed based on hybrid transforms and neuro fuzzy environment, i.e. in (Thakur, Dewangan, and Thakur 2014). The results obtained by the experiments demonstrate when performed comparison to the traditional techniques that include JPEG and JPEG2K, the compression technique suggested the minimized Mean Square Error (MSE). In (Thakur and Thakur 2014) a JPEG standard, fuzzy soft hybrid is proposed for deploying a constructive grey image compression codec. If compared traditional JPEG with JPEG2K standards the proposed fuzzy based soft hybrid JPEG technique has an enhanced ratio of compression, better and increased image quality.

2.2 Set Partitioning in Hierarchical Trees (SPIHT) Algorithm

In (A. Mallaiah 2012), for image compression with Huffman encoder SPIHT algorithm is proposed. For us to have a better image quality and enhanced definition, we have Retinex algorithm. Following a bit-plane sequence, Image wavelet transform coefficients of the image, consists of individual bits, that are coded with SPIHT. Recovering all the bits by coding transform (every single bit of it).

There is a perfect reconstruction by the wavelet transform, if infinite precision numbers is the storing of numbers as. perfect reconstruction only if its Possible scenarios include the practice of recovering the image consistent and perfect recovery after rounding. However, this is not the most efficient one.

We propose an integer multiresolution transformation, for lossless compression, we call it S+P transform, like the wavelet transform. During the transformation (instead of after) carefully truncating the coefficients of transform., it solves the finite-precision problem.

As we understand, complex compression algorithm tend to take more encoding time that compared to the decoding time. One of the most straight compression consequence because of the simplicity is the higher coding/decoding speed. The SPIHT algorithm is nearly regular or uniform, Coding time is equal to the time of decoding time.

2.3 Principal Component Analysis (PCA)

In (Seema Kalangi 2013), for the video compression, a technique called DWT is used. A 2D video file is converted from a 3D video, at the initial stages, for preventing the motion recompense step. PCA based video representation algorithm operations is enabled, visual change estimation and segmentation of video credits it by performance. In the PCA approach, the contained information which is a data set stored in a computational assembly with dimensions reduced that are anticipated upon by the integral projection of the data set onto a subspace produced by a system of orthogonal axes. Use of Singular Values Decomposition (SVD) method may help us obtain the optimal system of axes. Appropriate data features or properties are documented with some loss of data, it is achieved by reduced dimension computational structure.

3. TECHNIQUES TO EMBED

Used for the video frames , the embedding methods are classified into-

- Digital watermarking
- Data hiding algorithms

3.1 Digital watermarking

It is a procedure of approaching the digital signal into the digital content. The stages tangled in the digital watermarking procedure is represented in Fig.2

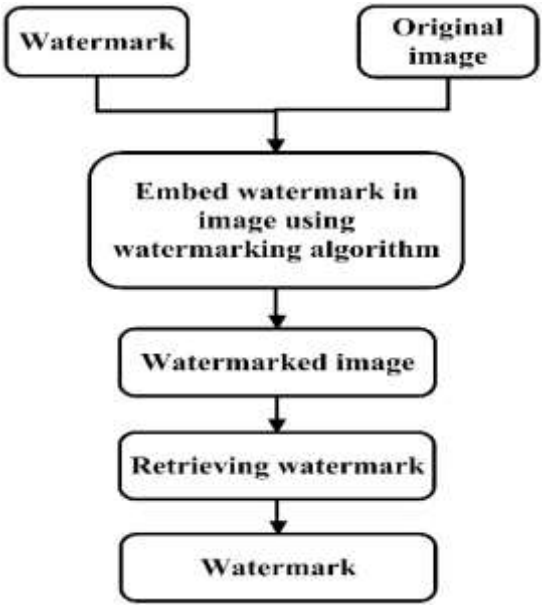


Fig2: Digital Watermarking overall process

At the beginning, the watermark image and the image that is original are embedded using the watermarking algorithm. Only after this process, the watermark image is obtained. While regaining the image the watermark and the original images are parted. An efficient digital watermarking must have the subsequent necessities,

- Strong
- Clear
- Ability to endure adjustments and misrepresentations.
- Effectual store and transmit

Fig.3 represents the classification of the digital watermarking techniques.

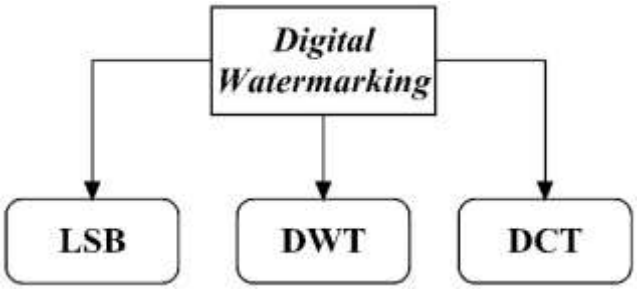


Fig3: Digital Watermarking Types

3.1.1 Least Significant Bit (LSB)

The information in the cover image is embedded with LSB approach. There are steps that are followed,

Steps involved in LSB

Step 1: Grey scale image is converted from a RGB image.

Step 2: Double correctness of the image is to be estimated

Step 3: In the watermarked image, most significant bits to be shifted to low significant bits

Step 4: Translate the least significant bits of the host image to zero.

Step 5: Modified host image is added by a shifted version of the watermarked image.

For the grey scale images, a LSB based digital image watermarking is anticipated, in (Chopra et al. 2012). Image embeds a message by exploiting the algorithm(LSB). Experimental results provide the proof of the watermarking algorithm, by demonstrating the optimal Mean Square Error(MSE) and Peak-Signal-to-Noise Ratio(PSNR) values. The image security is analysed using the LSB technique in (Singh, Shaw, and Aslam 2015). the influence of noise in

the images is also examined further. The examination of noise effect and image security shows the evidence of the presence of noise occurrence in the watermarked images, which presents an impact in the watermark image.

3.1.2 Discrete Wavelet Transform (DWT)

With a fixed window size, we have a time domain-based method for analysis, it is called DWT. There are four frequency bands, decomposed of the image processed by DWT. The bands are LL, LH, HL and HH. Low frequency districts include LL bands and LH, HL, HH are the high frequency districts. The sub-level frequency district information is produced by the application of DWT transformation for the low-level frequency component. According to (Senthil Nathan.M 2013) the 2D image after the application of three DWT disintegration is shown in Fig.4.There is a clear representation of low pass filter (L) and high pass filter (H). The frequency districts of the original image are HL1, LH1, HH1. The four districts such as LL2, HL2, LH2 and HH2 are decomposition of low-frequency district information. The original image signal information is placed in the frequency districts such as LH, HL, and HH. District information of the low-frequency is close to the original image.

Fig4: Decomposed image using DWT

<table>
<tr><td>LL3</td><td>HL3</td><td rowspan="2">HL2</td><td rowspan="3">HL1</td></tr>
<tr><td>LH3</td><td>HH3</td></tr>
<tr><td colspan="2">LH2</td><td>HH 2</td></tr>
<tr><td colspan="3">LH1</td><td>HH1</td></tr>
<tr><td colspan="3">LH1</td><td>HH1</td></tr>
</table>

According to (Shivani Khosla 2014), for the first average sub signal the accurate formula is depicted as follows,

$$a_n = \frac{f_{2n-1}+f_2}{\sqrt{2}}, n = 1,2,3,\ldots N/2 \quad (1)$$

where, N is the length of the signal. DWT based watermarking involves steps,

Step 1: Multiple sub bands using DWT are decomposed using original image.

Step 2: Necessary and valid sub band for watermark embedding are picked.

Step 3*:* Wavelet coefficients are picked on by the watermark image exploitation.

Step 4: Watermark image is obtained once the embedding process is complete.

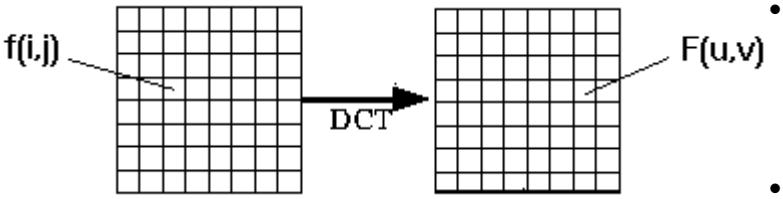
The scale parameter is discretized to integer powers of 2,2j,j=1,2,3.., in the discrete wavelet transform, it is to so the number of voices/octave remains 1. Difference between scales on a log2 scale is 1 for DWT. The conversion parameter is relative to the scale in the down sampled DWT. It means scale, 2j, translate by 2jm where m is a nonnegative integer. Scale parameter is limited within bounds to powers of two, in non-decimated discrete wavelet transforms (modwt and swt). Translation parameter is an integer.

Advantages of Discrete Wavelet Transform (DWT)

- The original signal can be easily reconstructed using inverse wavelet transformation
- We can use inverse wavelet transformation, to reconstruct the original signal.
- By disintegrating the original signal into wavelet transform, tends to provide the positional information.

3.1.3 Discrete Cosine Transform (DCT)

The spatial domain conversion of the signal is by exploitation of the cosine waveform, to the frequency domain. Image is represented as the sum of varying frequencies and magnitudes, using discrete cosine transform. The DCT supports to distinct the image into parts with respect to the visual class of the image, parts are also spectral sub-bands. The discrete cosine transform is very similar to the discrete Fourier transform. Transformation of a signal or an image from the spatial domain to the frequency domain. (Fig 7.8).



DCT Encoding

For a 1 dimension, we have the following general equation defined by the DCT:

$$F(u) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \Lambda(i).cos\left[\frac{\pi.u}{2.N}(2i+1)\right] f(i)$$

N data items,

and the corresponding ***inverse*** 1dimension DCT transform is simple $F^{-1}(u)$, i.e.:

where

$$\Lambda(i) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for} \xi = 0 \\ 1 & \text{otherwise} \end{cases}$$

The general equation for a 2 dimension (*N* by *M* image) DCT is defined by the following equation:

$$F(u,v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \Lambda(i).\Lambda(j).cos\left[\frac{\pi.u}{2.N}(2i+1)\right] cos\left[\frac{\pi.v}{2.M}(2j+1)\right] .f(i,j)$$

and the equivalent ***inverse*** 2 dimension DCT transform is simple $F^{-1}(u,v)$, i.e.:

where

$$\Lambda(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for} \xi = 0 \\ 1 & \text{otherwise} \end{cases}$$

The operation of the discrete cosine transform are :

The input image is N by M;

- The intensity of the pixel in row i and column j, f(i,j).
- The discrete cosine transform coefficient in row k1 and column k2 of the DCT matrix is F(u,v).
- Usually, for most images, in the upper left corner of the discrete cosine transform, we have much of the signal energy.
- Since the low rate values represent higher frequencies, often are small-small enough to be neglected with the little visible distortion, compression is achieved.
- Each pixel's gray scale level is contained in an array, i.e. the DCT input of an 8 by 8 array of integers.
- 8 bit pixels have levels from 0 to 255.
- 8 point DCT would be:

where

$$\Lambda(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for} \xi = 0 \\ 1 & \text{otherwise} \end{cases}$$

An image is represented as an image of sum of sinusoids of varying frequencies and magnitudes, with DCT.

The 2D DCT of an image is computed using dct2 function. For a typical image, maximum of the visually significant information about the image is focused in

just limited coefficients of the DCT. This is the reason, compression application for an image, include DCT. The international standard lossy image compression algorithm JPEG includes discrete cosine transform.

Advantages of DCT

- Stronger energy compaction property in DCT.
- Implementation is more efficient computationally.
- In the low-level of frequency components of DCT all the information is concentrated.
- Easy prevention of the high level frequency components.

3.2 Techniques For Data hiding

Data hiding, the process of using a typical cipher with an encryption key to encrypt the original image.

3.2.1 Data hiding of H.264/AVC video stream

A readable data-hiding algorithm is suggested for embedding the data, an efficient technique which quantizes into DCT coefficients, suggested in (Ma et al. 2010). We make use of a 4x4 DCT block, to address the misrepresentation introduced by the embedding process.

Using the intraframe prediction directions, distortion drift is being prevented. The vital distortion is decreased, and the embedding capacity is increased by using the algorithm. For the video stream H.264/AVC video stream, a scheme is proposed in (Xu, Wang, and Shi 2014).

Scheme includes components such as: H.264/AVC video encryption, data embedding, and data extraction. With the stream ciphers the residual coefficients are encrypted, with the properties of the H.264/AVC, the code words of the motion vector differences and intra-prediction modes, and for the code words for the residual coefficients. With the properties of the H.264/AVC, the code words of the intra-prediction modes, code words

H.264/AVC video-sequence based method is proposed in (Li, Chen, and Zhao 2010), for data hiding. The suitable data is hidden using the appropriate quantization and transformation coefficients. We make use of desirable data recovering process to recover the data hidden. Further, from the encoded stream they are extracted.

3.2.2 Data hiding techniques for MPEG video

The proposed method exploiting, solves the video bit stream size increment related issues. There are two approaches proposed for the compression of MPEG video, in (Shanableh 2012), two data hiding approaches. The quantization scale of the constant bitrate video is our first approach for hiding data. The exploitation of the macro block collation for hiding the message bits is our second approach. The macro blocks are assigned to the arbitrary slice groups by exploitation of the content of the image. With minimum distortion and compression overhead, the proposed approaches produce an efficient message payload.
Thus, the proposed methods yield optimal message payload with slight distortion and compression overhead.

4. PROPOSED WORK

Fig.4. represents the formation framework of the block code. We make use of the framework to embed the successive video frames.

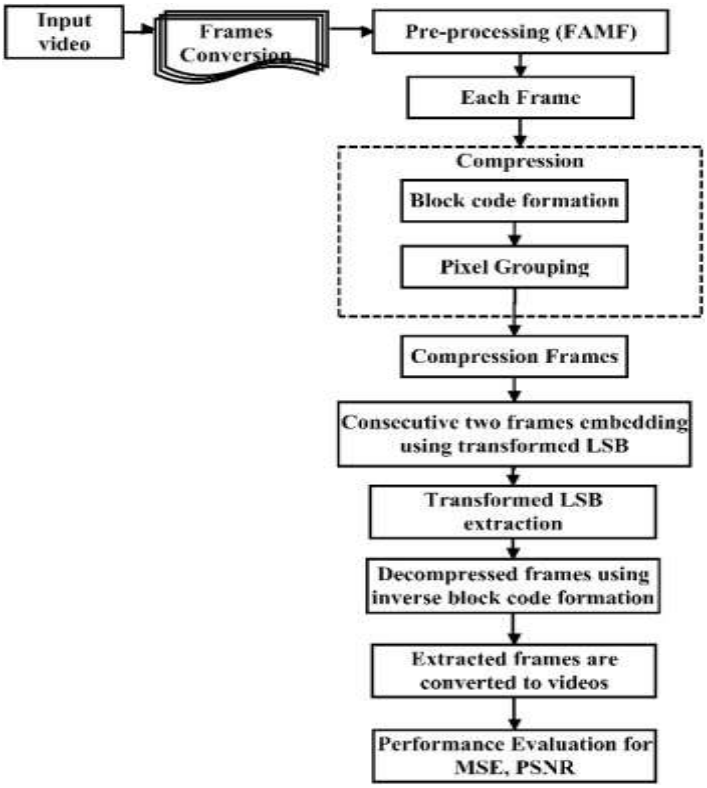


Fig 5: Block code formation framework

Multiple split of the input video at first, Fuzzy Adaptive Median Filter (FAMF) will process each frame. Noises will be deducted in the preprocessing. Compression using block wise pixel grouping takes place before embedding the frames. The splitting of the image into patches and estimates the recurrent pixels and location for all patches, is done using the suggested compression process. Before the pixel values, estimated pixel values will be placed. Code takes output of each patch is considered. The compressed frames are embedded by the embedding techniques that include Least Significant Bit (LSB), once all the frames are compressed. Using inverse block code the decompression of the compressed frames done ate the receiver end. Videos are then converted back from the decompressed frames.

4.1 What is Python zlib?

For using the DEFLATE lossless compression algorithm, we have a Python interface to the zlib library of C, Python zlib library. We can make use of the library commercially; the format of the compression is open source in nature and has not been fallen under any patent. It is quite portable and completely lossless in nature. There is no expand in the data at all, with respect to the mechanism.

There are two application on major level, compression and decompression, whether be it a structured in-memory content, or a string, or a file.

Library includes features such as compression and decompression. Operation including both compression and decompression can be done as a one-off operation, or by splitting of the data into portions. In UNIX system we make use of gzip file format/tool, it is based on DEFLATE.

It is also well-matched (compatible) with the gzip file format/tool (which is also based on DEFLATE), used in UNIX systems.

We can compress a string of data by making use of the zlib library, that provides us with the functionality of compression.

Taking only two arguments, we have syntax of this function:

compress (data, level=-1)

The bytes that are to be compressed are contained in the argument data. Level takes an integer value, range to -1 or 0 to 9. The value or the level of compression is determined by this parameter, level 1 is yields the lowest level and is fastest. Highest level of compression is by level 9, it is the slowest.

The default is level 6 which is represented by the value -1. There is a balance between speed and compression with the value -1. There is no compression yielded by level 0. We make use of the compressobj() function to manage large data streams, which return a compression object.

Following is the syntax:

compressobj (level=-1, method=DEFLATED, wbits=15, memLevel=8, strategy=Z_DEFAULT_STRATEGY[, zdict])

The two functions compressobj() and compress(), has differences in between, the wbits arguments, the window size controller, window size is controlled by it, is the difference, apart from the data parameter(whether the header and trailer are included in the output.). The algorithm used for compression is represented by the method argument. The only method defined in the RFC 1950, the only possible current value is DEFLATED. Compression tuning is related to the strategy argument.

4.2 File Compression

Function compress() to compress the data in a file.

Image file, we will try to compress, a "pic1.png" (make sure image is in the folder same as python file).

Below is the code:

```
import zlib
o_data = open('pic1.jpg', 'rb').read()
compressed_data = zlib.compress(o_data, zlib.Z_BEST_COMPRESSION)
ratio = (float(len(o_data)) - float(len(c_data))) / float(len(o_data))
print('Compressed: %d%%' % (100.0 * ratio))
#o_data is original data
#ratio is compressed ratio
#c_data is compressed data
```

Constant Z_BEST_COMPRESSION is being used in the code above, in zlib.compress(..) line. This algorithm has to offer this level of best compression. We need to calculate the ratio of the length of the compress data to length of the original data.

Output-

```
$ python compress_file.py
Compressed: 8%
```

In order to compress the mp4s, we make use of ffmpy library. ffmpy is a Python library that provides access to the ffmpeg command line utility. ffmpeg is a command-line application that can perform transformations on video files, including video compression, which is the most commonly requested feature of ffmpeg.

ffmpeg -i input.mp4 -vcodec libx264 -crf 20 output.mp4

The name of the input file is specified by the -i clause that proceeds ffmpeg.

After the input file name, the command specifies output options. In this case, the output options specify the video codec to use and the constant rate factor, or CRF, to use to compress the video file. The CRF, which, in this case, is 20, can range from 18 to 24, where a higher number will compress the output to a smaller size. The last part of the instruction is the name of the resulting compressed file.

If our input and output file names are stored in variables input_name and output_name, then we could set up a dictionary called inp to store the parameters to apply to the input file, and a dictionary called outp to apply to the output file. Here is the code:

```
inp={input_name:None}
outp = {output_name:'-vcodec libx264 -crf %d'%crf}
```

We pass these dictionaries to the FFmpeg function defined in the ffmpy library to create an FFmpeg object.

```
ff=ffmpy.FFmpeg(inputs=inp,outputs=outp)
ff.run()
```

For debugging purposes, it might be handy to see what corresponding command-line instruction the FFmpeg object is executing.

Code to take one mp4 file and compress it using a user-specified constant rate factor:

```
import ffmpy
input_name = input("Input file name: ")
crf = int(input("Enter constant rate factor between 18 and 24: "))
output_name = input("Output file name: ")
inp={input_name:None}
outp = {output_name:'-vcodec libx264 -crf %d'%crf}
ff=ffmpy.FFmpeg(inputs=inp,outputs=outp)
print(ff.cmd) # just to verify that it produces the correct ffmpeg command
ff.run()
print("done!")
```

5. Conclusion

During the transmission, for giving an efficient data transfer and data security maintenance, many techniques are exploited that include video compression and embedding. This paper includes various compression techniques for image, techniques like SPIHT, fuzzy concepts and PCA based methods. These techniques are analyzed and studied. There are algorithms such as digital watermarking, digital embedding and data hiding, each are discussed in detail. There is loss of pixel information during the transformation, compressed images are not efficiently restored by the compression techniques already existing. We don't receive any satisfying image or video security, the computational work complexity and time complexity, increases in the traditional compression algorithm. We propose a framework to consider all into account, all the issues related, with a block code formation. There is an exploitation of block wise pixel grouping technique, it is done to perform the compression.

The framework includes all the codes and algorithms needed for the lossless compression. Python libraries that are already existing and making work with frames and video rates in the background can increased in terms of the efficiency. There is a splitting into multiple patches of the images involved in the process of compression. For each of the patch, location (recurrent) of the pixel is found. Prior to the pixel value, the estimated locations of each pixel is placed, i.e. for the entire image. We make us of the LSB algorithm to perform the embedding process, after each frame has been compressed. MSE, PSNR, SSIM, are few of the metrics that will determine the superiority and the relevance of the framework proposed.

All the above algorithms do not contain in any python framework, there are libraries that can be imported to perform specific tasks, we intent to provide a whole new study.

6. References

[1] Mallaiah, S. K. Shabbir, T. Subhashini. 2012. "An Spiht Algorithm With Huffman Encoder For Image Compression And Quality Improvement Using Retinex Algorithm." *International Journal Of Scientific & Technology Research* no. 1 (5):45-49.

[2] Chopra, Deepshikha, Preeti Gupta, Gaur Sanjay, and Anil Gupta. 2012. "LSB based digital image watermarking for gray scale image." *IOSR Journal of Computer Engineering (IOSRJCE) ISSN*:2278-0661.

[3] Deb, Kaushik, Md Sajib Al-Seraj, Mohammed Moshiul Hoque, and Md Iqbal Hasan Sarkar. 2012. Combined DWT-DCT based digital image watermarking technique for copyright protection. Paper read at 7th International Conference on Electrical & Computer Engineering (ICECE).

[4] Gurpreet Kaur, Kamaljeet Kaur. 2013. "Image Watermarking Using LSB." *International Journal of Advanced Research in Computer Science and Software Engineering* no. 3 (4):858-861.

[5] Kashyap, Nikita, and GR Sinha. 2012. "Image watermarking using 3-level discrete wavelet transform (DWT)." *International*

Journal of Modern Education and Computer Science no. 4 (3):50.

[6] Khalilian, Hanieh, and Ivan V Bajic. 2013. "Video watermarking with empirical PCA-based decoding." *IEEE Transactions on Image Processing* no. 22 (12):4825-4840.

[7] Khan, Asifullah, Ayesha Siddiqa, Summuyya Munib, and Sana Ambreen Malik. 2014. "A recent survey of reversible watermarking techniques." *Information sciences* no. 279:251-272.

[8] Li, Y., H. x. Chen, and Y. Zhao. 2010. A new method of data hiding based on H.264 encoded video sequences. Paper read at IEEE 10th International Conference on Signal Processing (ICSP), 24-28 Oct. 2010.

[9] Ma, Xiaojing, Zhitang Li, Hao Tu, and Bochao Zhang. 2010. "A data hiding algorithm for H. 264/AVC video streams without intra-frame distortion drift." *IEEE Transactions on Circuits and Systems for Video Technology* no. 20 (10):1320-1330.

[10] Navnidhi Chaturvedi, Dr.S.J.Basha. 2012. "Comparison of Digital Image watermarking Methods DWT & DWTDCT on the Basis of PSNR." *International Journal of Innovative Research in Science, Engineering and Technology* no. 1 (2):147-153.

[11] Pan, I-Hui, Ping Sheng Huang, and Te-Jen Chang. 2013.

[12] "DCT-Based Watermarking for Color Images via TwoDimensional Linear Discriminant Analysis." In *Information Technology Convergence*, 57-65. Springer.

[13] Sadashivappa, K.V.S Anand Babu, Dr. Srinivas 2011. "Color Image Compression using SPIHT Algorithm." *International Journal of Computer Applications* no. 16 (7):34-42.

[14] Seema Kalangi, Veeraiah Maddu, Sreenivasa Ravi

[15] Kavuluri. 2013. "A Novel Approach of Low Complexity DWT/PCA Based Video Compression Method." *International Journal of Engineering Science and Innovative Technology (IJESIT)* no. 2 (3):159-168.

[16] Senthil Nathan.M , Pandiarajan.K, Baegan.U. 2013. "Digital Image Watermarking Basics " *IOSR Journal of*

[17] *Electronics and Communication Engineering (IOSRJECE)* no. 8 (1):07-11.

[18] Shanableh, T. 2012. "Data Hiding in MPEG Video Files Using Multivariate Regression and Flexible Macroblock Ordering." *IEEE Transactions on Information Forensics and Security* no. 7 (2):455-464. doi: 10.1109/TIFS.2011.2177087.

[19] Shivani Khosla, Paramjeet Kaur 2014. "Secure Data Hiding Technique Using Video Steganography and Watermarking." *International Journal of Computer Applications* no. 95 (20):7-12.

[20] Singh, Ranjeet Kumar, Dilip Kumar Shaw, and M Javed Alam. 2015. "Experimental Studies of LSB Watermarking with Different Noise." *Procedia Computer Science* no. 54:612-620.

[21] Suganya.G, Mahesh.K. 2014. "A Survey: Various Techniques of Video Compression." *International Journal of Engineering Trends and Technology (IJETT)* no. 7 (1):10-12.

[22] Suhail, M. A., and M. S. Obaidat. 2003. "Digital watermarking-based DCT and JPEG model." *IEEE Transactions on Instrumentation and Measurement* no. 52 (5):1640-1647. doi: 10.1109/TIM.2003.817155.

[23] Thakur, S, Nilesh Kumar Dewangan, and Kavita Thakur. 2014. A Highly Efficient Gray Image Compression Codec Using Neuro Fuzzy Based Soft Hybrid JPEG Standard. Paper read at Proceedings of Second International Conference “Emerging Research in Computing, Information, Communication and Applications” ERCICA.

[24] Thakur, Vikrant Singh, and Kavita Thakur. 2014. Design and Implementation of a highly efficient gray image compression

codec using fuzzy based soft hybrid JPEG standard. Paper read at International Conference on Electronic Systems, Signal Processing and Computing Technologies (ICESC).

[25] Wong, KokSheik, Kiyoshi Tanaka, Koichi Takagi, and Yasuyuki Nakajima. 2009. "Complete video qualitypreserving data hiding." *IEEE Transactions on Circuits and Systems for Video Technology* no. 19 (10):14991512.

[26] Xu, Dawen, Rangding Wang, and Yun Q Shi. 2014. "Data hiding in encrypted H. 264/AVC video streams by codeword substitution." *IEEE Transactions on Information Forensics and Security* no. 9 (4):596-606.

[27] Ray Klump, Professor and chair of Mathematics and Computer Science Director, Master of Science in Information Security Lewis University, http://www.lewisu.edu/experts/wordpress/index.php/compressing-mp4-files-listing-directory-contents-and-copying-files-to-a-remote-server-in-python/

[28] Scott Robinson, https://stackabuse.com/python-zlib-library-tutorial/