AUTHOR: - VIGNESH VASUDEVAN

PROJECT- MPEG H264 VIDEO COMPRESSION

AIM: -

The aim of this experimentation is to see what the MPEG H264 compressions for the video compression performs and how the compression algorithm works majorly on a video compression. I have chosen a few videos for sample purpose. However, the performance is little lagging in MATLAB for a higher resolution and longer videos.

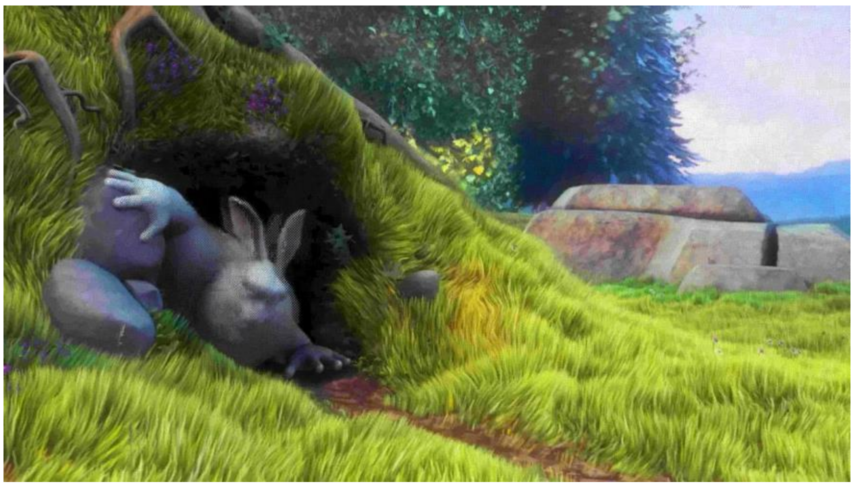
Chroma Subsampling.

- The video is converted into frames using the Video Reader function. Once the video is converted into frames, I have performed the below actions.
- I have performed the chroma subsampling technique for converting the RGB image to YUV format.

Intra Encoding

- The intra Encoding is just encoding the I frame into a 4*4 frame from the respective height and width of the original frame.
- We now perform the 4*4 Integer transform in H264.
- Three modes of encoding are performed in the Intra frame.
- The I frame gets blurry pixels than the P and B frames, while using the integer transformation. This might be a case of study for further experimentation.

I frame: -



Inter Encoding

- The inter encoding is the P and B frame.
- 8*8 frame is used for motion estimation.

- I have used the logarithmic search window for the motion vectors and finding the minimum error that the frame is providing.

P Frame

B Frame

Motion Estimation

- In the implementation of Motion Estimation, I have performed the logarithmic search in an 8*8 frame.
- We take the current frame and the reference frame for comparing the difference of motion between the two frames and calculate the error that is supposed to occur during the motion estimation.
- I have calculated the row and column traverse across each frame and find the absolute difference between the motion to calculate the error frame.
- We subtract the error from the current frame across every traverse and hence try to achieve a lossless error during this process.

Integer & Inverse Integer Transform

- I have created a 4*4 matrix for multiplying the below values by calculating the integer transform Cf.
- Quantization matrix Mf is created and integer transformation is performed.

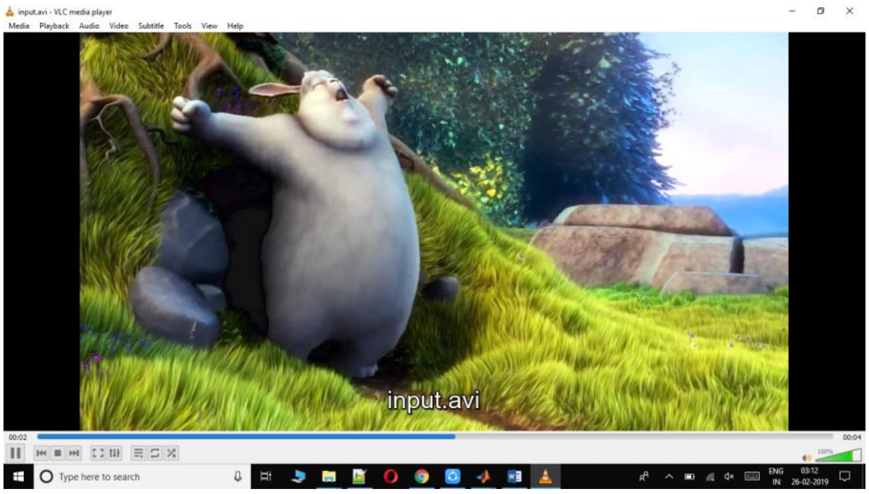
$$Y = \text{round}\left([C_f]\times[X]\times[C_f^T]\cdot M_f/(2^{15})\right).$$

- Inverse Integer Transformation is performed using the below formula. The same 4*4 matrix is reconstructed. Dequantization Vi table is also created

$$Z = \text{round}\left([C_i^T]\times[Y\cdot V_i]\times[C_i]\,/(2^6)\right).$$

Sample Video used for testing and experimentation purpose

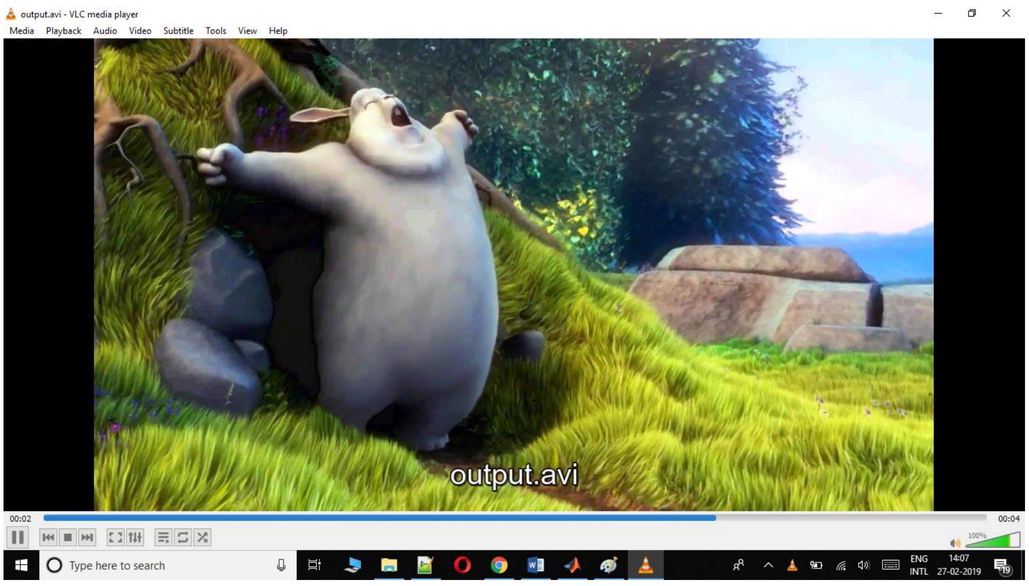
Video frame-1280*720


input.avi - VLC media player
Media Playback Audio Video Subtitle Tools View Help
input.avi
00:02
00:04
Type here to search

Sample constructed input I, P, B frames

Sample reconstructed output I, P, B frames

Sample reconstructed Video

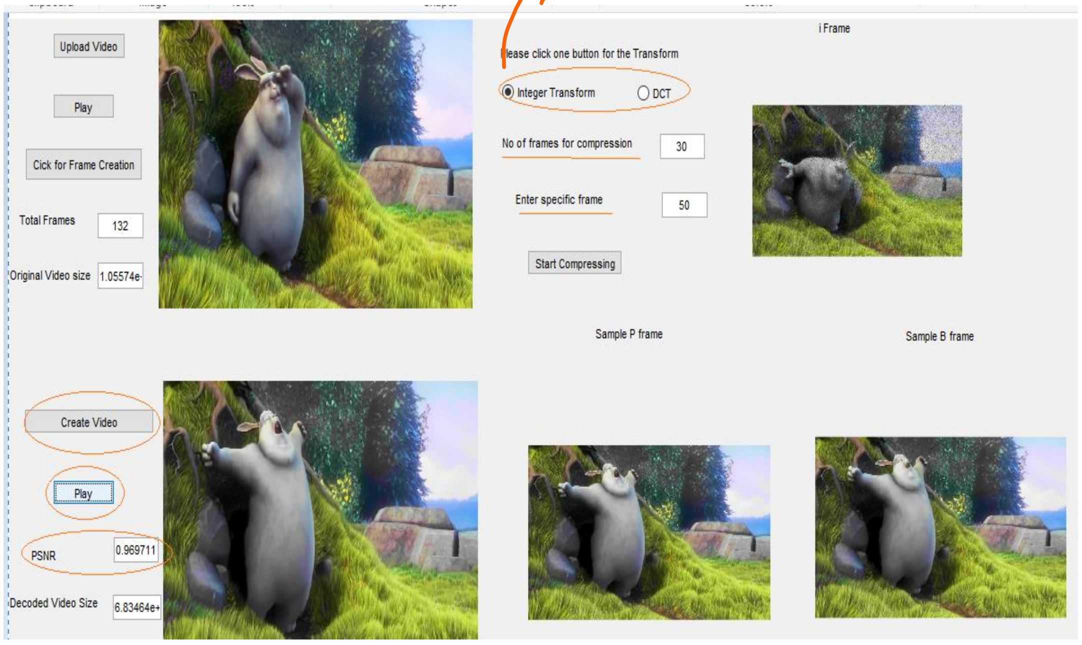

output.avi - VLC media player
Media Playback Audio Video Subtitle Tools View Help
output.avi
00:02
00:04
100%
Type here to search
ENG INTL
14:07
27-02-2019

GUI: -

I have created a GUI which provided an overview of the Video compression and enhancements.

The Upload Video and Play pushbutton is created such that it uploads the video from the local drive and plays the video.

I have provided options to perform both the DCT and Integer transform option so that the user can select the type of transformation needed for the video



Total Frames 132

Original Video size 1.05574e-

Please click one button for the Transform

[✓ Integer Transform] DCT

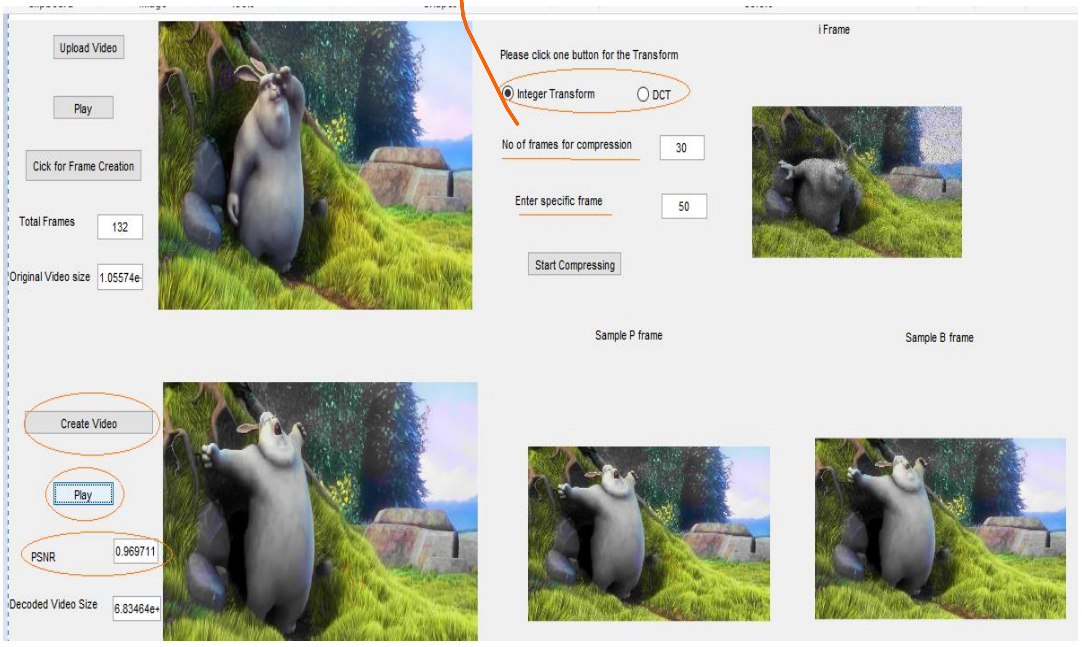
No of frames for compression 30

Enter specific frame 50

PSNR 0.969711

Decoded Video Size 6.83464e+

I have created the GUI in such a way that it's customizable and we can choose the number of frames we need to perform the video compression.



Total Frames 132

Original Video size 1.05574e-

Please click one button for the Transform

[✓ Integer Transform] DCT

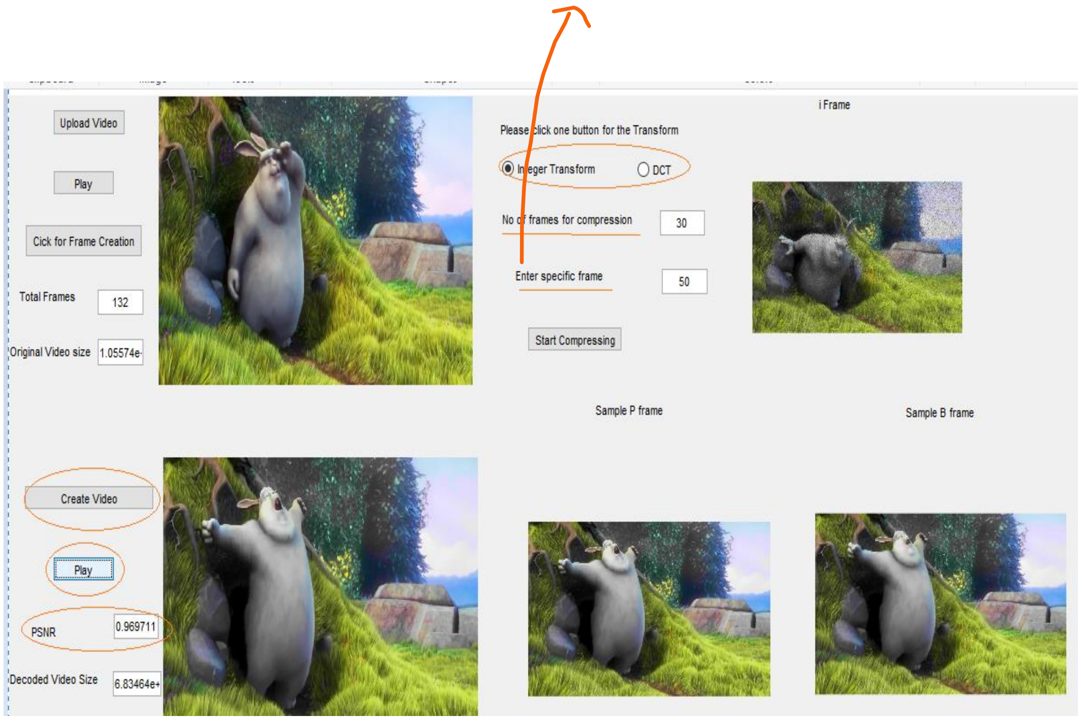
No of frames for compression 30

Enter specific frame 50

PSNR 0.969711

Decoded Video Size 6.83464e+

The user can also provide the specific frame from which the video encoding can occur. Suppose if we have a huge video and we need only a specific part for compressing, then this option is highly useful. I have started compressing the video from the 50th frame until 80th frame.



Total Frames: 132
Original Video size: 1.05574e
Please click one button for the Transform: [✓ Integer Transform] DCT
No of frames for compression: 30
Enter specific frame: 50
PSNR: 0.969711
Decoded Video Size: 6.83464e+

Scaling and Quantization

- We are performing a 4*4 Quantization and scaling after the Integer transformation.
- The future purpose is to give the user the facility to try both DWT, DCT and Integer transform and see the loss curve.
- Here I have tried the Integer transformation, hence the quantization loss is calculated for every frame. The total PSNR value is calculated after every transformation.

For Sample above video 1

Frame	MSE	PSNR
I frame	165.032	25.955
P frame	99.46	28.15
B Frame	87.366	28.71
After 50th frame		Total PSNR=0.5743

- I tried for various experiments and various factors to check on the working of H264 encoding. While trying for another sample video, I could see different results with different experimentations.

Results for another sample video

Frame	MSE	PSNR
I frame	130.50	26.96
P frame	97.13	28.14
B Frame	78.02	28.67
After 132nd frame		Total PSNR=0.219

However, MPEG H264/H265 is a vast area for video encoding with various algorithms for prediction and lossless compression. Hence this can be further explored and need to see the experimentation results.