Video Plus Depth Compression for Mobile 3D Services

Abir Ahmed, Dumkene Izuora, Ferhan Jamal, Tobi Ajila and Vincent Chukwuekezie

Abstract- This report is on the implementation of a paper on video plus depth compression for optimization of 3D mobile services. The limitation of mobile services such as bandwidth demand and low memory prompted the authors to propose an efficient compression formats for 3D mobile services. In their paper, they evaluated the two different MPEG coding standards: MPEG-C part 3 and H.264 auxiliary picture syntax, using four different video scene content and complexity. The experimental result showed that an efficient compression for mobile 3Ds can be obtained at lower bitrate.

In this paper, the overview of 3D video, MPEG C part 3 and H.264 AVC standard were given. The works done in the original paper were implemented and the results obtained were compared with the results given in the original paper. It was found that the results were similar and recommendations and conclusions were made based on the experiment.

Keywords— mpeg-c part 3, H.264 AVC, video plus depth, compression

I. INTRODUCTION

This paper is on the implementation of the paper written by P. Merkle, Y. Wang, K. Muller, A. Smolic and T. Wiegand which was titled “ video plus depth compression for mobile 3D services”. It was published in 3DTV conference: the true vision-capture, transmission and display of 3D video, in 2009 by IEEE. In the referenced paper the 3D depth feature of the 3Dtv was reviewed and the two different H.264 based coding methods for video and depth, MPEG-C part 3 and H.264 auxiliary picture syntax, were evaluated. Their simulation was done using JM 14.2 implementation of the H.264/MPEG-4 AVC codec. The motivation for the experiment was to reduce the complexity of some 3D videos for mobile applications by compressing them.

II. OVERVIEW OF 3D VIDEO

The 3D video has information about the volume and depth of a scene unlike the 2D video. It can be represented using any video format but whatever format used must be able to produce a means for stereoscopic viewing, since this is what gives the depth perception of the visual information of the 3D video [1].

The stereoscopic representation of the 3D video can be done using the video plus depth technique [1]. The video plus depth technique is the production of 3D video from 2D video. In this technique two separate 2D video signals are used to provide color image information and the depth which is associated to each pixel. The depth is the distance of the images in the scene from the camera. The values of the depth are usually represented as integers which are in the range of 0 to 255. With 8 bits per pixel the depth will result in a gray-scale depth map. The video plus depth format has high complexity than the stereo view because it needs either additional computation in order to get the depth values from multiple views in a scene or a special image acquisition hardware which can get the depth map directly from a scene [1].

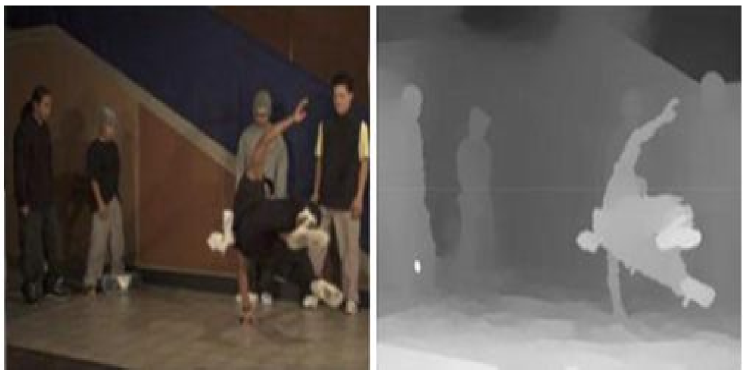


Fig 1: Video (left) plus depth (right) [1]

III. MPEG C PART 3

MPEG-C Part 3, also known as ISO/IEC 23002-3 standard, basically specifies the representation of auxiliary video and supplementary information [1]. This terminology gives the representation format for depth maps which in-turn encode them as conventional 2D sequences with existing video compression standards. MPEG-C Part 3 is based on the encoding of 3D content as video plus depth [4]. The basic requirements fulfills by this technology are as follows [4]:

a. Interoperability
b. Display technology
c. In-dependency
d. Backward Compatibility and compression efficiency

The overview diagrams of MPEG-C Part 3 are shown below with H.264/AVC encoders generating two streams (BS), one for video and another for depth. Before encapsulating into a single transport stream (TS), the two streams are multiplexed . The streams are de-multiplexed at the receiving end and the output video and depth signals are produced by independently decoding which in turn are used to generate the second stereo view [4].

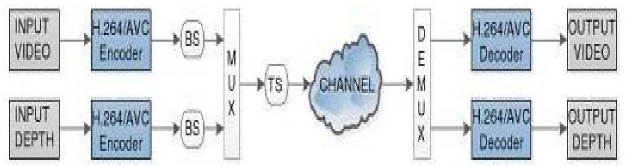


Fig 2: Schematic Block Diagram of MPEG-C Part 3 [4]

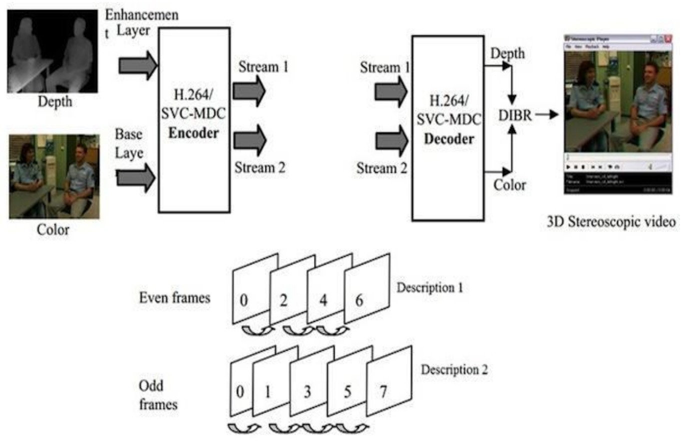


Fig 3: Scalable multiple-description coding scheme for video-plus-depth format [5]

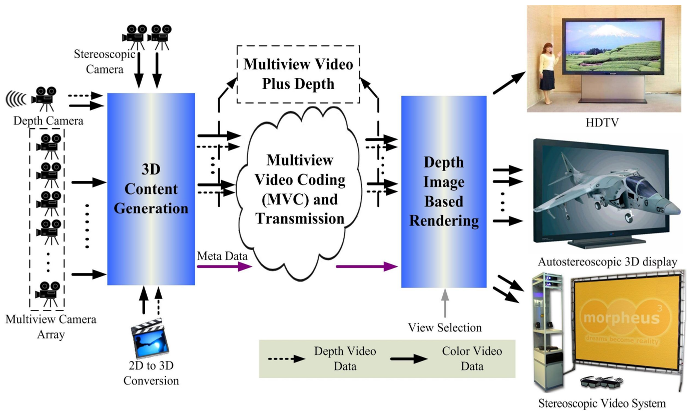


Fig 4: Multiview plus depth (MVD) coding and transmission [6]

A transmission system and general coding for MVD is depicted in figure 4. The depth information can be estimated from the video signal itself provided by special range cameras or solving for stereo correspondences while a few cameras are necessary to acquire multiple views of the scene. At the receiver (depth image based rendering); we are getting the auto-stereoscopic 3D displays, Stereoscopic video system, etc. because of the depth and stereoscopic images of the source.

In figure 2 and figure 3, using video plus depth as the exchange format allows [7]:

a. Backwards compatibility with 2D.

b. Independency regarding the display and the capture technology.

c. Compatible with most "2D to 3D" algorithms.

d. Compression efficiency is excellent.

e. User-controlled global depth range.

This approach also has some limitations which are as follows [7]:

a. It is capable of handling limited depth range as occlusions areas are very big.

b. The stereo signals are not easily accessible by this format.

c. The things obtained are not 100% accurate.

To conclude, we can say that MPEG-C Part 3 provides an easy and efficient way for attaching depth values to an image. It allows using existing video compression standards, while its flexible structure makes it easily upgradeable. In addition, the advantages of video plus depth approach are fully exploited or accomplished. As explained above the advantages of MPEG-C Part3 are various which is simplicity, low 3D overhead, display independency, backwards compatibility with 2D, adjustable depth-effect at display, etc.. Lastly, a sample for mobile applications has been developed, which demonstrates the efficiency of such a method.

IV. H.264/AVC STANDARD

The H.264/AVC standard was built on the concepts of MPEG-2 and MPEG-4 visual [2]. It has the potential for better compression efficiency.

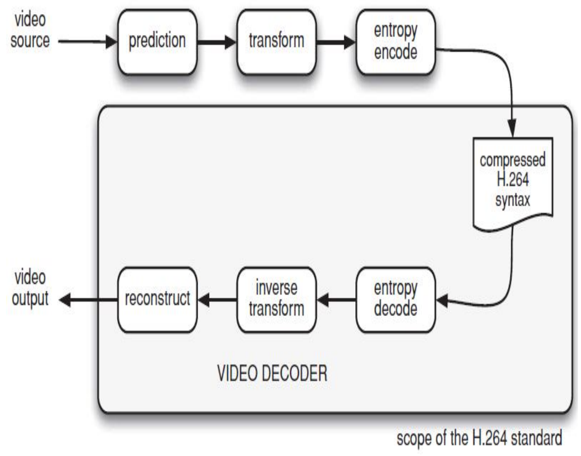


Fig 5: The H.264 video coding and decoding process. [2]

The H.264 video encoder performs prediction, transformation and encoding to produce compressed H.264 bitstream, while its decoder decodes, inverse transforms and reconstructs the bitstream to produce decoded video sequence [2].

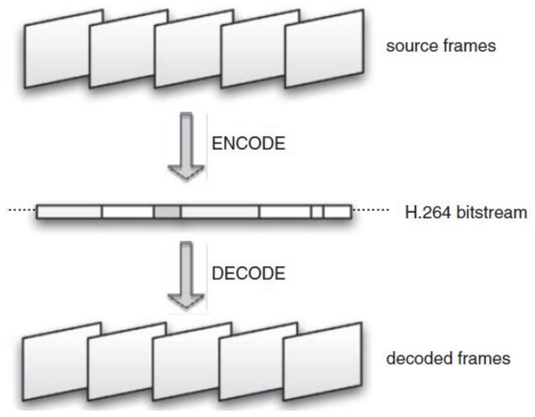


Fig 6: source frames, bitstream and decoded frames. [2]

The H.264 is a lossy compression format and because of this its decoded video sequence is not identical to the original video sequence [2].

In the H.264 codec the video data is processed in units of a macroblock (MB) which correspond to 16 by 16 display pixels. In the encoder, a residual macroblock is formed by generating a prediction macroblock and subtracting it from the current macroblock. The residual macroblock is transformed, quantized and encoded. In parallel, the quantized data are rescaled, inverse transformed and then added to the prediction macroblock which is used to reconstruct the coded version of a frame and stored for later predictions. In the decoder, the macroblock is decoded, rescaled and inverse transformed to form the decoded residual macroblock [2].

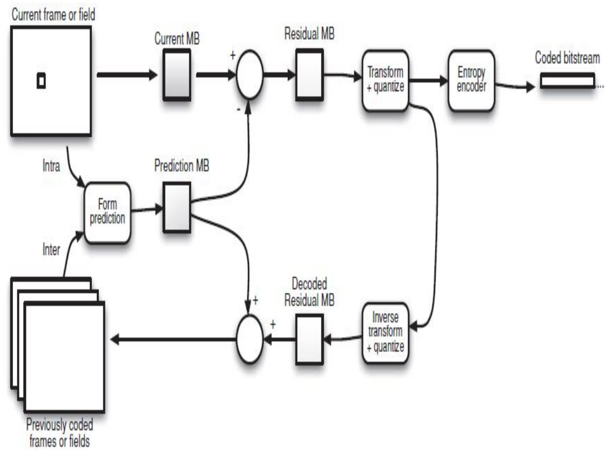


Fig 7: Typical H.264 encoder [2]

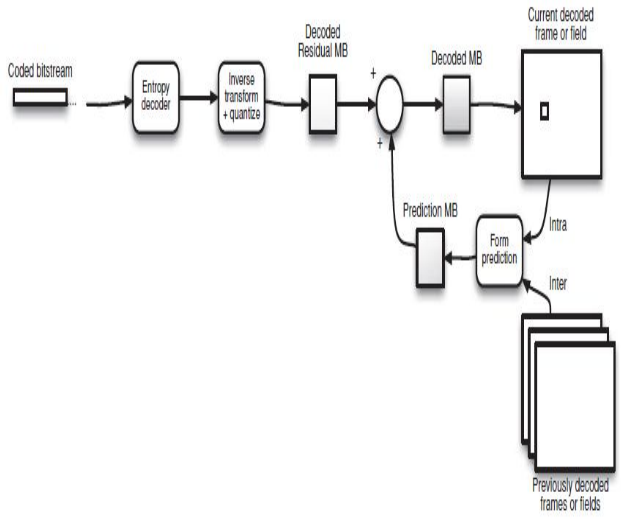


Fig 8: Typical H.264 decoder [2]

The prediction of a current macroblock is formed by the encoder based on previously coded data, which can be either from the current frame using the intra prediction or from other frames that were already coded and transmitted using inter prediction [2].

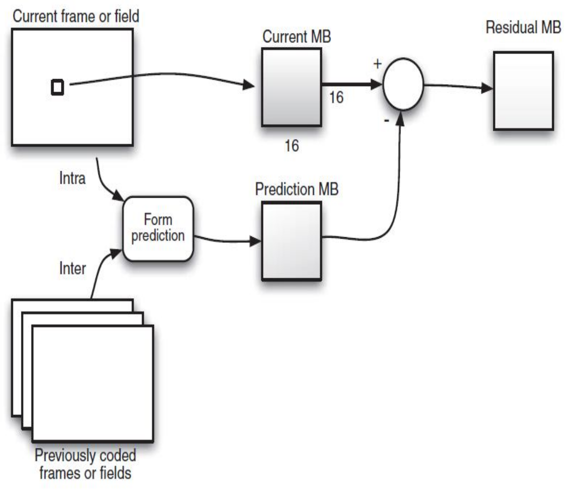


Fig 9: Prediction flow diagram. [2]

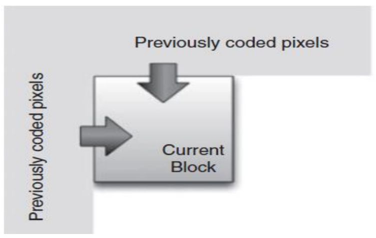


Fig 10: intra prediction [2]

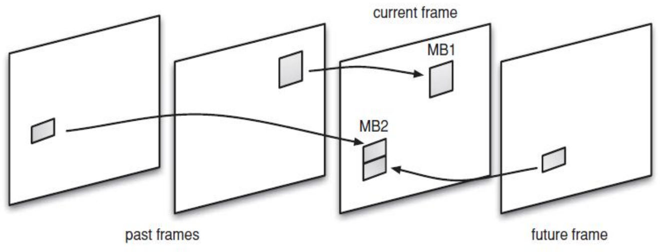


Fig 11: Inter prediction [2]

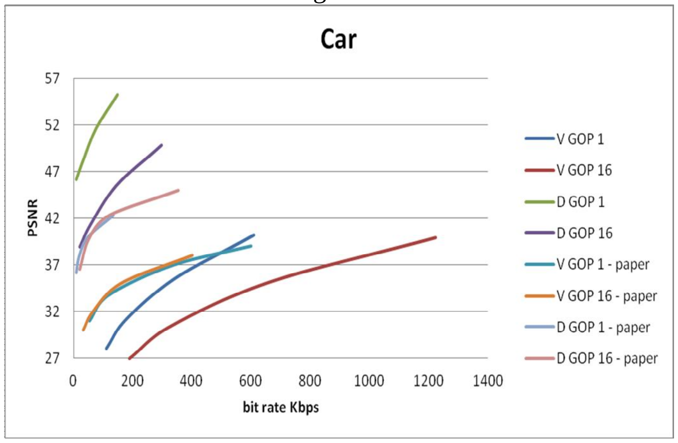
V. Experiments

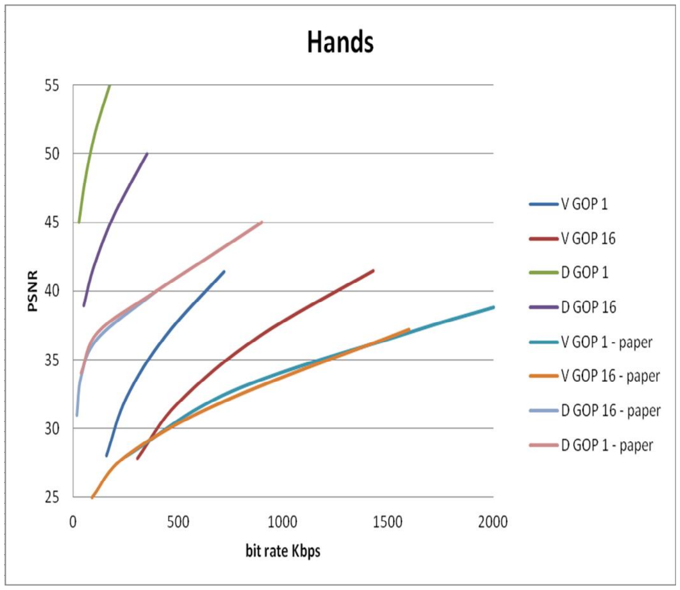
The authors of the paper performed simulations that attempt to display the performance of video and depth encoding in realistic conditions for mobile applications. The purpose of the experiments was to understand what the transport bit-rate requirements are to achieve acceptable viewing quality, and which factors determine this.

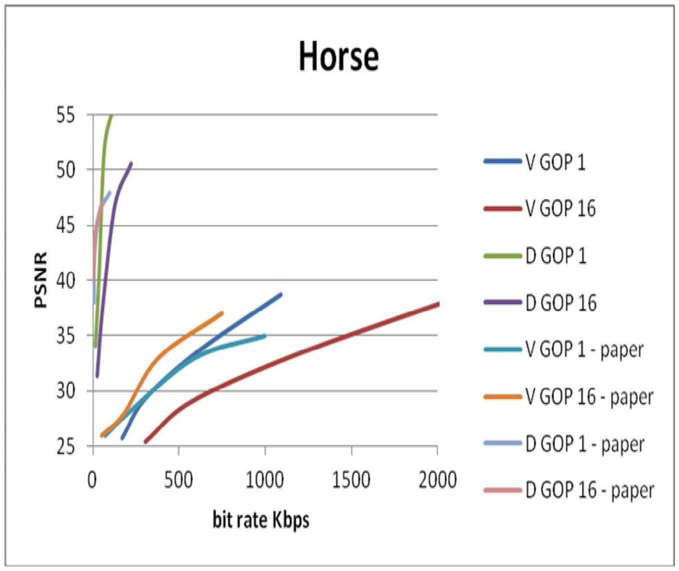
The paper experiments with the Mpeg C part 3 encoding scheme using JM 14.2 to supply the H.264/AVC encoder/decoder implementation. For the first set of experiments JM was configured with the following parameters: intra period of 16 frames, this specifies the max period of I-coded frames in an encoded sequence. A search range of 32, which sets the allowable search range for motion estimation. Four quantization parameters (QP = 24, 30, 36, 42) which determines the quality of the pictures. Finally, two temporal predictions structures, GOP size 1 for simple structures and GOP size 16 for hierarchical B structures.

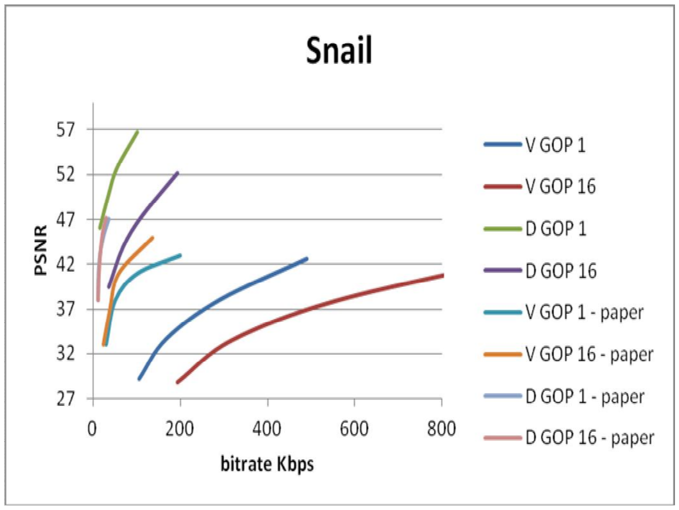
The coding scheme was simulated with four short videos each about 5 - 10 seconds long. The first is a car driving along a road, the second a person washing their hands, the third a horse grazing on a farm and the fourth a snail with a black background. All of the videos chosen are significantly different from each other with the intention of understanding the performance of this coding scheme with different video content. All the videos had a frame rate of 30fps with a resolution of 480x270 pel.

The results were the following:





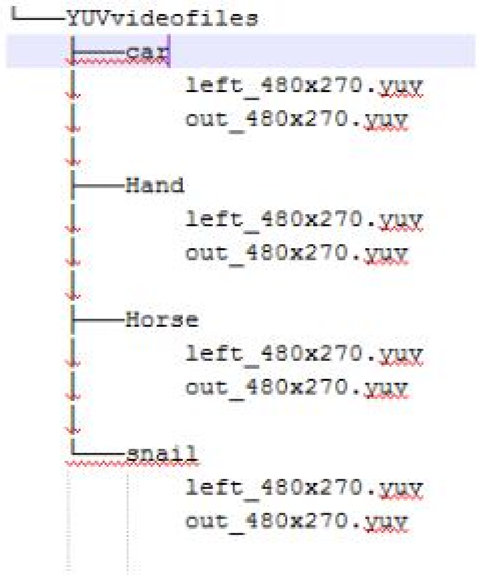




VI. Implementation

The implementation requires the use of JM 14.2 to gather the necessary data to confirm the findings from the paper. The JM 14.2 project was compiled in Visual Studio. In order to gather the necessary data to compare with the results in the paper, we automated a significant portion of the process to gather the necessary data.

Automation was done using Python. The Python script takes the following three arguments:
location to the JM binary that was built using Visual Studio('--JM'), the path to the JM configuration file that will be used('--config'), and the directory of where the raw YUV videos are to be found('--yuv-directory'). The ''--yuv-directory'' argument pointing to a directory can contain multiple directories in it where each directory contains the raw YUV files of both the left eye footage and the depth footage. The following shows the directory structure graphically:



```
└──YUVvideofiles
    ├──car
    │       left_480x270.yuv
    │       out_480x270.yuv
    │
    ├──Hand
    │       left_480x270.yuv
    │       out_480x270.yuv
    │
    ├──Horse
    │       left_480x270.yuv
    │       out_480x270.yuv
    │
    └──snail
            left_480x270.yuv
            out_480x270.yuv
```

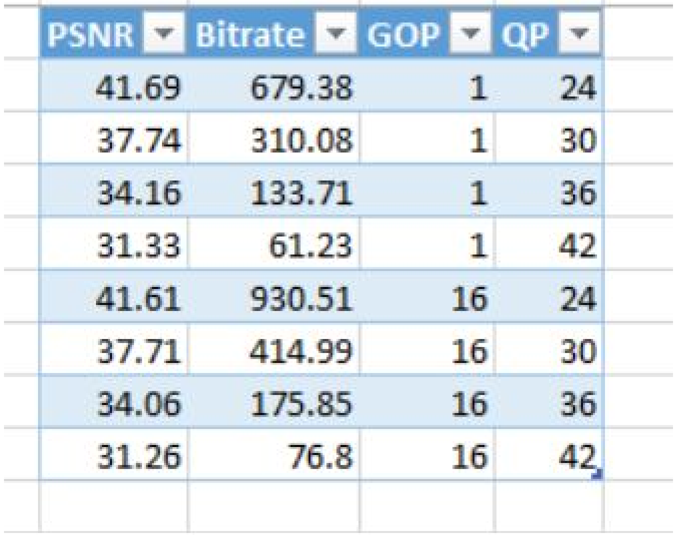
This allows the script to be extensible by simply creating a new directory under the directory supplied to the '--yuv-directory' argument and rerun the script which will run the tests for all the YUV footage. The script will walk the directory and assemble a list of files to run the tests against. As well, the JM binary can easily be specified so the tests can be re run identically for a different JM version.

The script will run the JM binary multiple times with dynamically changing configuration between each run for every YUV footage in the subdirectories of '--yuv-directory'. The changing configuration between runs is based on the information gathered from the paper as well as directly from one of the authors of the paper that was contacted to clarify some uncertainties about the work done by them. This dynamic configuration was implemented with a Python dictionary which is just a hashtable(key value pairs) of all the dynamic parameters. Adding new parameters to dynamically change is as simple as adding an entry to the 'DynamicOptions' dictionary in the script and a call to setListValue definition between each run. A separate configuration file is used to specify the static JM configuration that will not change between runs (passed as '--config'). The script expects the filename of the YUV footage in the subdirectory to be "left_480x270.yuv" for the left footage and "out_480x270.yuv" for the depth footage, ignoring other files so as to not try to encode other data that might be in those directories.

After each test, the output from the JM binary is parsed to gather the appropriate data using Regular Expressions. After grabbing the relevant data through regex, a XML file is generated using a schema created by us as an easy way to represent the data. The XML file follows the pattern where there are multiple <Entry> elements where an <Entry>

represents a single run with particular parameters. There are child elements inside <Entry> containing different metrics for the particular run. For every YUV video, a new xml is created with the video name and is put in the same location is the source YUV file with the ".xml" extension. This XML can then be used to analyze the data.

To take advantage of the XML, it can be imported into Microsoft Excel. Excel will intelligently interpret the XML and populate the cells as well as headings for the columns. We can then use the data in the spreadsheet to create graphs and make our analysis. The resulting spreadsheet looks like this after importing the XML for one of the raw YUV images:



PSNR	Bitrate	GOP	QP
41.69	679.38	1	24
37.74	310.08	1	30
34.16	133.71	1	36
31.33	61.23	1	42
41.61	930.51	16	24
37.71	414.99	16	30
34.06	175.85	16	36
31.26	76.8	16	42

This can then be repeated for every XML file generated where each XML file corresponds to multiple encodes with different parameters of one YUV file. The instructions to import XML into Excel is available in the README supplied with the script.

The implementation above provides the infrastructure to automate encoding and gather necessary data. This paper requires significant number of runs which may not be feasible running manually to gather the data.

VII. Analysis

The results show that the bitrate required for acceptable viewing varies based on the content of the video, the GOP and depth complexity. It is evident that GOP 16 encoding configuration yields lower bitrates for the same viewing quality.

Our results do not exactly match that of the papers, but for the most part the analysis is the same. We observe that GOP 16 pictures require a lower bit rate for the same quality.

VIII. Feedback

The paper targets mobile applications for its research but perhaps the term "streaming" applications is more suitable since the goal of the video plus depth coding scheme is to reduce the transport bitrate required for acceptable video quality. Certainly, this would apply to mobile applications like cell phones and tablets, but this would also apply to desktop PCs, smart TVs and other network media players.

One area that we suspect can be improved in terms of the compression capabilities is the depth data. We postulate that the depth data is not as essential as the 2D video in terms of accuracy. A slightly distorted 2D image is noticeable but a slightly distorted depth is easily overlooked. Currently, for every pixel of the 2D image there is a monochrome pixel of depth data. To improve the compression one could down sample the depth data by factor, similar to chroma subsampling in perceptive coding schemes.

An area of struggle for us was understanding the experiment performed by the authors of the paper. It was not immediately clear how they actually performed the experiment and what the results meant. It was only after investigating the JM14.2 application that we understood how they got their results and the implications of their results. The paper describes some of the parameters that were required but not the full list, we had to contact the writers of the paper in order to get results that somewhat matched the paper. The paper does not specify if the PSNR is for the Y channels, the chroma channels or an average of all the channels.

IX. Conclusion

This paper explores ways to make 3D video more feasible for mobile applications. By encoding video and depth data as opposed to the commonly used stereoscopic video source, one can greatly reduced the bitrates required for optimal viewing. Our experiments show that the use of hierarchical B (GOP 16) pictures improves the bit rate required for viewing. We found the paper to be very informative and we were able to produce similar results. We think the paper could have done a better job of describing how they performed the experiments. In closing, we find this topic to be very interesting and can envision video plus depth playing a part in future encoding schemes for 3D viewing.

REFERENCES

[1] P. Assuncas, L. Pinto and S. Faria. " 3D media representation and coding" in 3D future internet media. Springer New York. 2014. Pp. 9-36

[2] I.E Richardson. "What is H.264?" in The H.264 advanced video compression standard. 2nd ed. John Wiley and Sons Ltd.2010. Pp. 82-91

[3] M. Cagnazzo, B. Pesquet-Popescu and F. Dufaux. "3d video representation and formats" in emerging technologies for 3D video creation, transmission and rendering. John wiley and sons Ltd. 2013. Pp. 113.

[4] [4] C.W Chen, Z. Li and Lian. " Representation and coding formats for stereo and multiniew video" in intelligent multimedia communication: techniques and applications. Springer-Verlag Berlin Heidelberg. 2010. pp. 58

[5] 2D/3D Video Processing and Coding : http://www.surrey.ac.uk/cvssp/activity/labs/i-lab/2d3d_video_processing_and_coding.htm

[6] http://codec.siat.ac.cn/yunzhang/projects/3DVSystem.jpg

[7] A. Bourge, J. Gobert and F. Bruls, "MPEG-C part 3: Enabling the introduction of video plus depth contents," in Proc. of IEEE Workshop on Content Generation and Coding for 3D-television, Eindhoven, The Netherlands, June 2006.