MP4 File Compression with FFMPEG

Daniel Wilson

Abstract

Improvements in video resolution and color mean more hard drive space is required to store the necessary data, increasing the importance of minimizing the storage requirements of video files. The MPEG file format uses keyframes to lessen storage requirements by representing frames as changes from previous frames instead of building the frame from scratch. One problem that currently exists is that it is difficult to determine the optimal number of keyframes to use. Adding keyframes is computationally intensive, and their effectiveness at lowering storage requirements is very inconsistent. In this paper, we perform experiments to evaluate the effectiveness of keyframes in different types of videos, to find the optimal number of keyframes in various situations. We measure the compression ratio for different videos at varying keyframe values, and the time it takes to compress them using FFMPEG. We show that how much a video benefits from keyframes depends on the image content of the video. Finally, we demonstrate that we can apply this knowledge to improve MPEG codecs to better calculate the optimal number of keyframes for various videos.

Motivation

Hard drive space and network speeds are both limited resources, and it is therefore important to use as little of them as possible. By minimizing the space in which video files are stored, we can lessen the amount of these resources that are used.

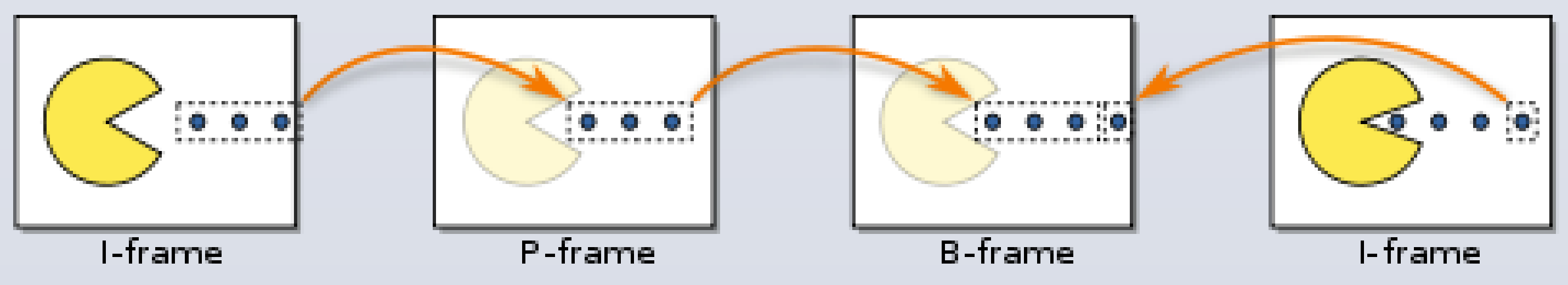
Keyframes and Motion Compensation

I-Frames are independent frames. They have all the information necessary for an image.

P-Frames are also known as predictive frames. They represent their frame based on changes in information from the previous frame.

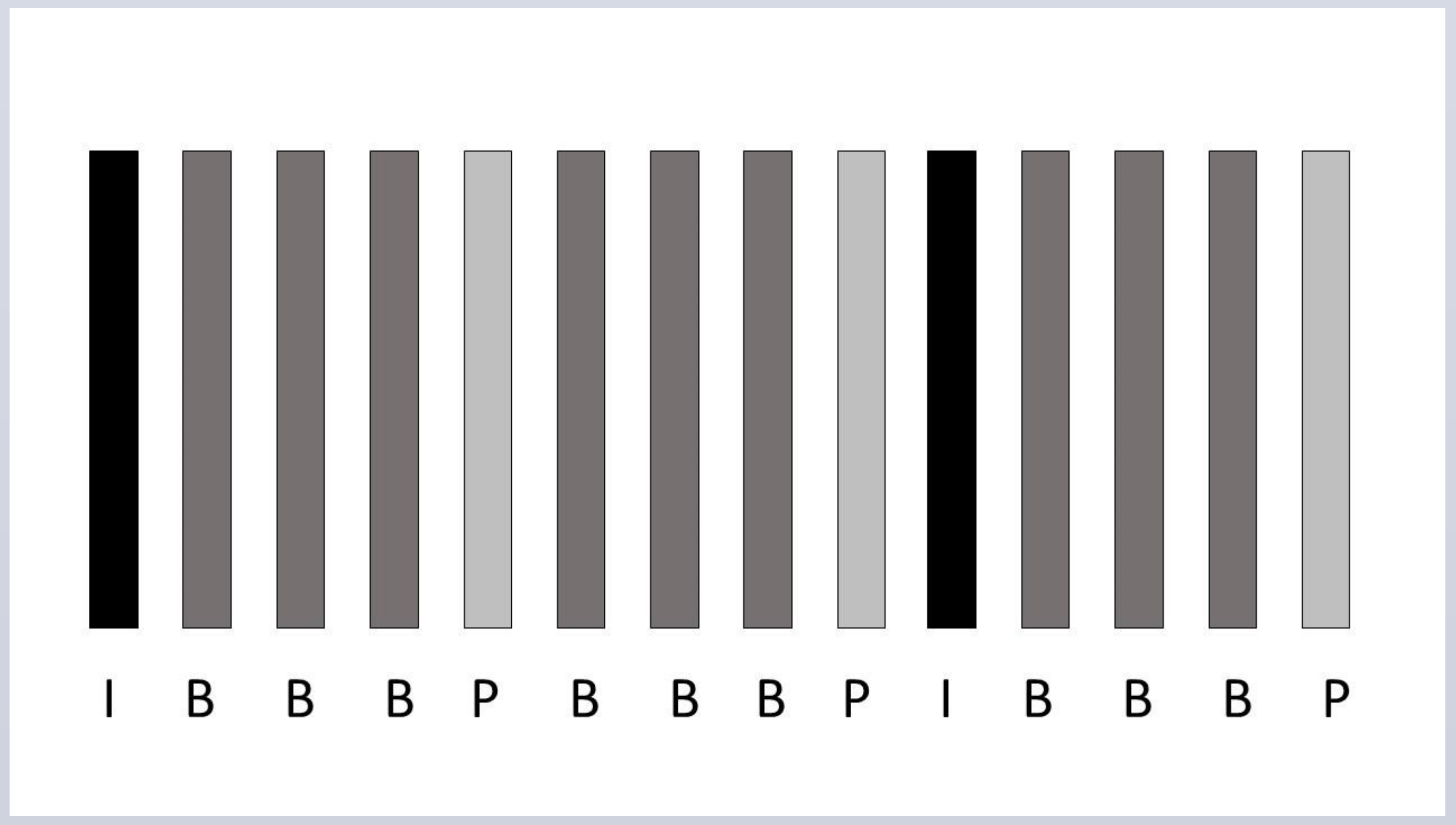
B-Frames, or bidirectional predictive frames, use information from the previous and next frames to represent the current frame.

P and B frames are also referred to as keyframes.



https://en.wikipedia.org/wiki/Video_compression_picture_types

Every MP4 file contains its own sequence of different frame types that make up the video, an example of which is shown in the following image.

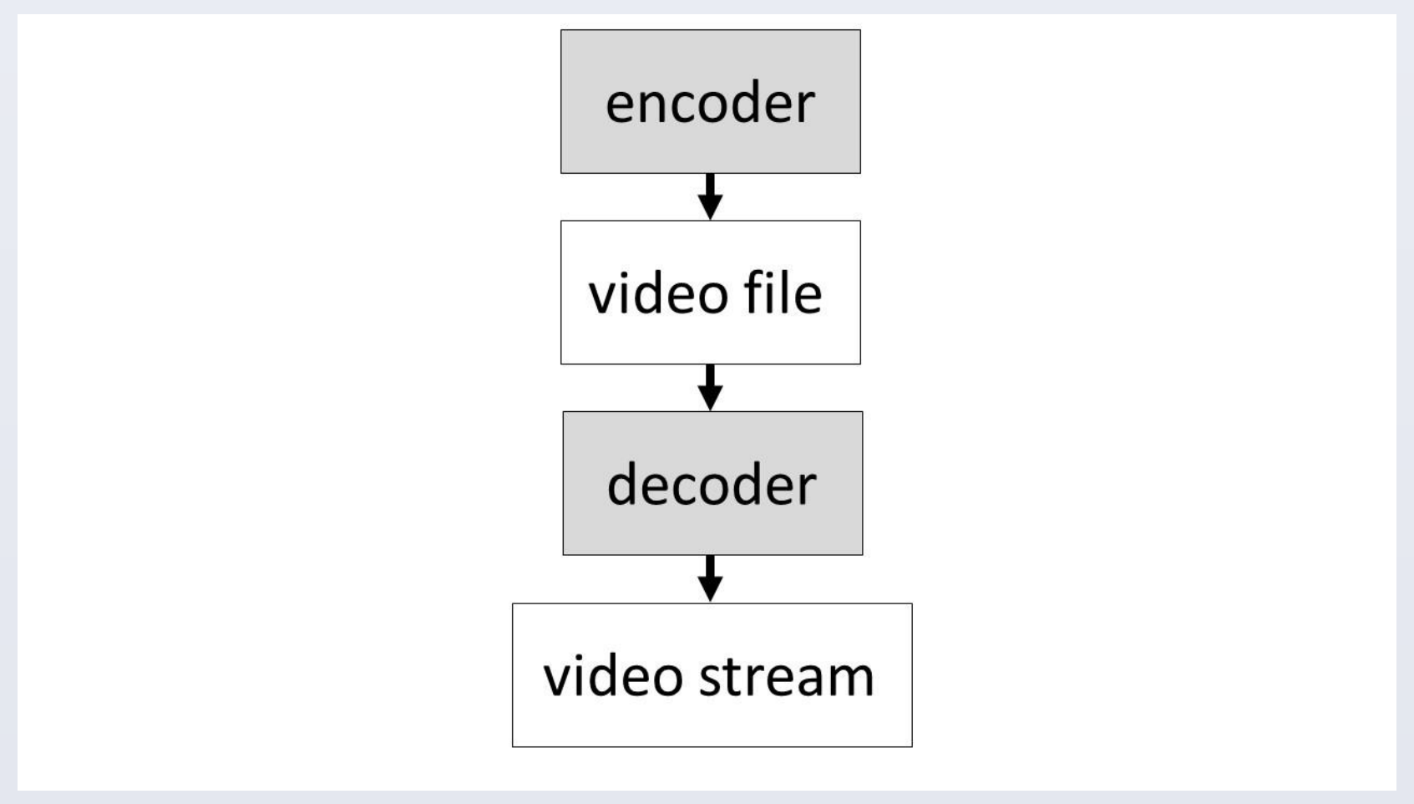


[8]

Design

To accomplish our goal, we have designed an experiment with FFMPEG where we will test videos to find their optimal allocations of keyframes. We have chosen six sample videos which vary dramatically in file size, resolution, color, and length. We will conduct tests where we compress the sample video files using controlled values for the number of consecutive keyframes. With this information, we will make charts showing how the compression ratio and the time required to compress a file depends on the number of consecutive keyframes used. With this data, we will draw conclusions about how to optimally allocate keyframes for different types of videos. Finally, we will use this information to modify a codec to better allocate keyframes based on the results we find from this experiment.

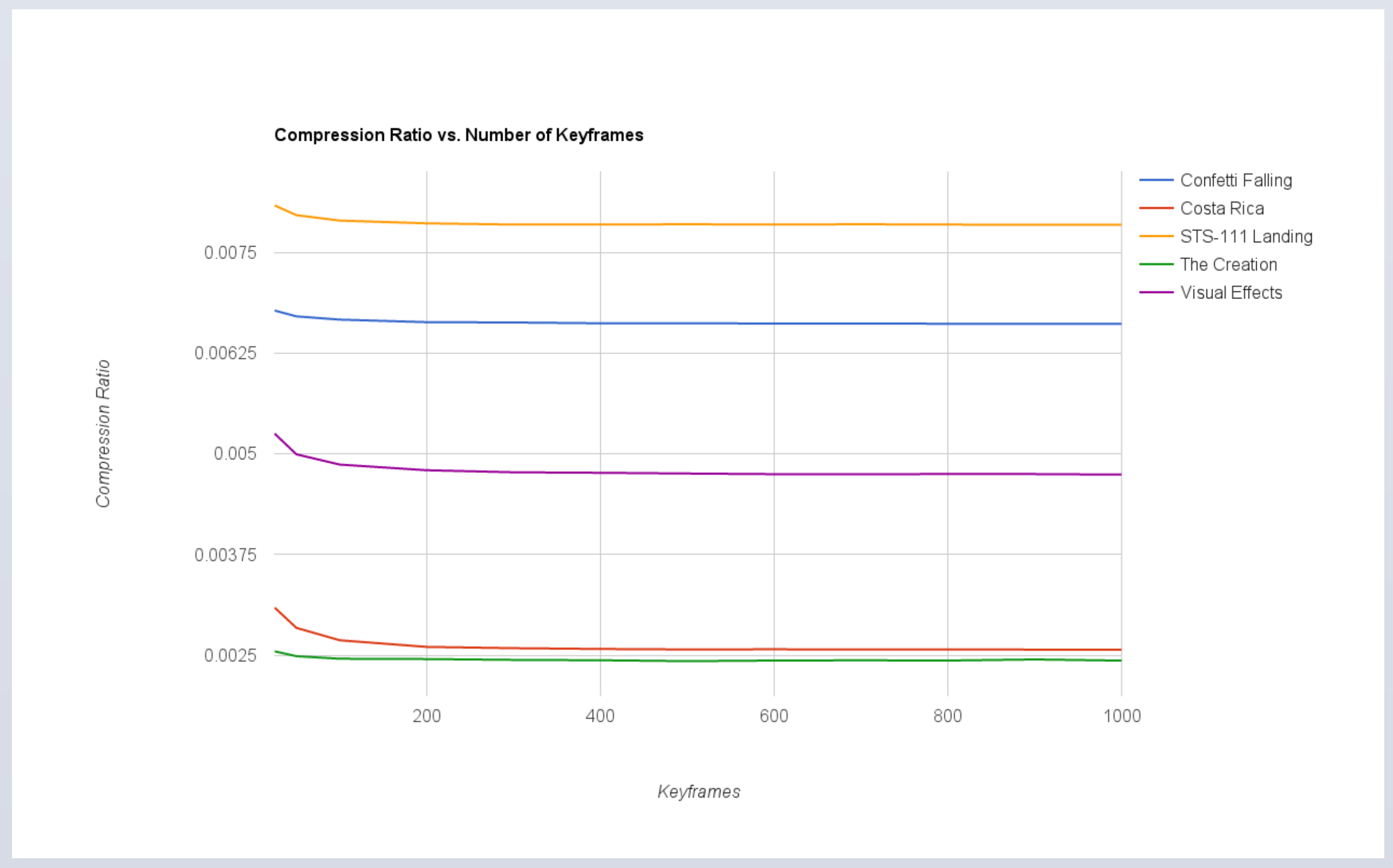
The software that we use is called FFMPEG, which is a free, open source, command line software which compresses MP4 files. We will use the h264 codec for compressing the files. The diagram below displays the relationship between the encoding and decoding software and the video.

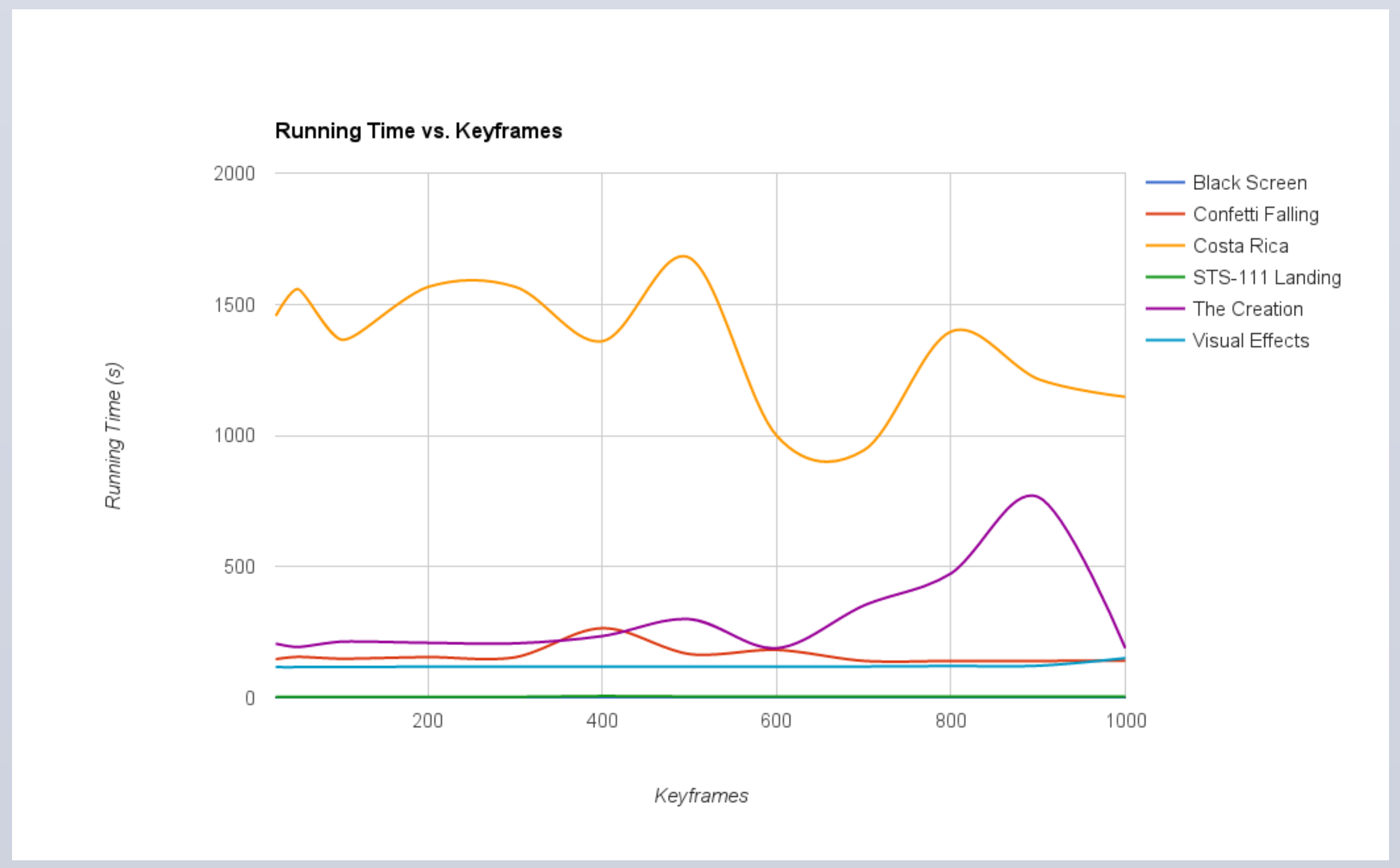


$$\begin{tikzcd}[row sep=small] \text{encoder} \arrow[d] \\ \text{video file} \arrow[d] \\ \text{decoder} \arrow[d] \\ \text{video stream} \end{tikzcd}$$

Preliminary Results

When conducting our tests, we recorded the relationship between the number of keyframes used and the compression ratio. We then placed this information in a table to represent the relationship between the number of keyframes and ratio of compression of each video tested. Next, we recorded the run time to encode the video at each number of keyframes, and compiled this information into the graphs displayed below.





Conclusions

The relationship between the number of consecutive keyframes used and the compression ratio seems to vary based on the type of video being compressed. The videos of Costa Rica and the spaceship landing, for instance, are both real-world videos captured with a video camera. They both seem to have the best compression ratios at around 400-500 keyframes. After this, there was little to no benefit to adding more consecutive keyframes to the video file. Interestingly, this rule seems to hold true in spite of the fact that the two videos contain completely different resolutions. This allows us to draw two conclusions. First, typical video footage from cameras compresses best at around 400-500 keyframes. Second, the resolution of the video does not seem to impact the optimal number of keyframes.

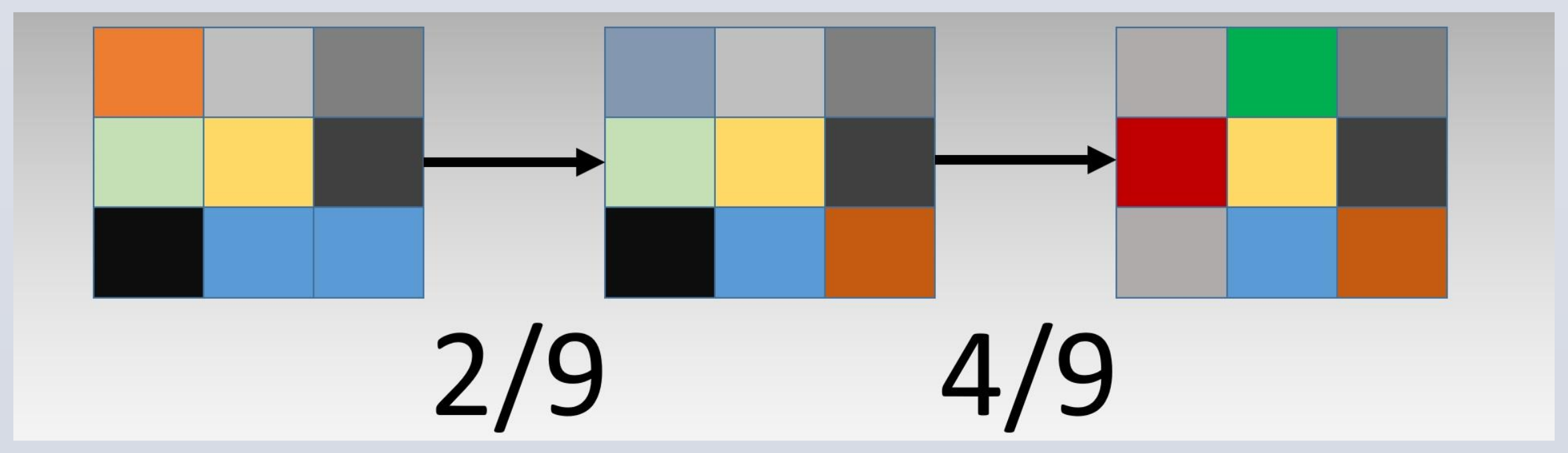
The video of the black screen had large size reductions when compressing the video to include up to 1000 consecutive keyframes. This suggests that videos with consistent color and simplistic images benefit a particularly large amount from the use of keyframes.

The videos of the confetti falling and visual effects only seemed to benefit from increasing the number of keyframes out to around 250 frames. Due to the unpredictable changes in the video as the confetti falls, it is more difficult to represent this information using keyframes. Thus, the video has less reduction in file size from increased use of keyframes.

Using the conclusions described above, we can approximate the optimal number of keyframes for various types of videos, as is shown in the table below.

Video Type	Example	Optimal Keyframes
Consistent Color	Black Screen	1000
Average Color Change	Video Camera Footage	500-600
Random Color Changes	Confetti Falling	200-250

Now that we have a better idea of how many keyframes to allocate for different types of video, we will modify FFMPEG to allocate specific amounts of keyframes to certain parts of the video based on the type of footage. Videos that benefit the most from keyframes contain consistent colors. The more the colors change from one video to the next, the less beneficial keyframes are to the file size. Therefore, we have designed an algorithm that compares macroblocks between frames and finds which fraction of them have changed. Based on the change between frame, we can allocate an appropriate number of keyframes when encoding the video. We are currently working on incorporating this algorithm in FFMPEG.



Acknowledgements

Xunfei (advisor)

FFMPEG: https://www.ffmpeg.org/

Thanks to Baptiste Coudurier, Carl Eugen Hoyos, Lou Logan, and all the other developers of FFMPEG.

References

[1] B. Fazzinga, S. Flesca, F. Furfaro, and E. Masciari. Rfid-data compression for supporting aggregate queries. ACM Trans. Database Syst., 38(2):11:1–11:45, July 2013.

[2] S. Gringeri, R. Egorov, K. Shuaib, A. Lewis, and B. Basch. Robust compression and transmission of mpeg-4 video. In Proceedings of the Seventh ACM International Conference on Multimedia (Part 1), MULTIMEDIA '99, pages 113–120, New York, NY, USA, 1999. ACM.

[3] J. Katto and M. Ohta. Mathematical analysis of mpeg compression capability and its application to rate control. In Proceedings of the 1995 International Conference on Image Processing (Vol.2)-Volume 2 - Volume 2, ICIP '95, pages 2555–, Washington, DC, USA, 1995. IEEE Computer Society.

[4] D. Le Gall. Mpeg: A video compression standard for multimedia applications. Commun. ACM, 34(4):46–58, Apr. 1991.

[5] D. A. Lelewer and D. S. Hirschberg. Data compression. ACM Comput. Surv., 19(3):261–296, Sept. 1987.

[6] U. Manber. A text compression scheme that allows fast searching directly in the compressed file. ACM Trans. Inf. Syst., 15(2):124–136, Apr. 1997.

[7] R. Mantiuk, A. Efremov, K. Myszkowski, and H.-P. Seidel. Backward compatible high dynamic range mpeg video compression. ACM Trans. Graph., 25(3):713–723, July 2006.

[8] K. Mayer-Patel, B. C. Smith, and L. A. Rowe. The berkeley software mpeg-1 video decoder. ACM Trans. Multimedia Comput. Commun. Appl., 1(1):110–125, Feb. 2005.

[9] K. Patel, B. C. Smith, and L. A. Rowe. Performance of a software mpeg video decoder. In Proceedings of the First ACM International Conference on Multimedia, MULTIMEDIA '93, pages 75–82, New York, NY, USA, 1993. ACM.