Data Compression Techniques for MPEG-4 Files

Daniel Wilson
Computer Science,
Earlham College,
Richmond, Indiana,
dlwilson13@earlham.edu

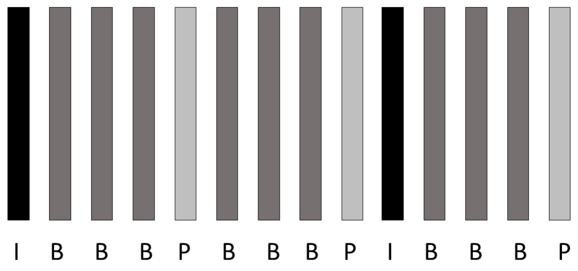
1. INTRODUCTION

As better technology becomes available, the quality of videos increases. Resolution, frame rates, and color options can all be improved to provide a better user experience. These advancements come at the cost of data, since higher quality video will require more storage space. This means video files will take up more space on hard drives and take longer to send over network connections. The motivation for this project is to combat these issues by examining existing data compression algorithms for MPEG-4 files and attempting to improve them to further reduce the space required to store data. While many file formats and codecs have been researched as methods of compressing video files, this project will focus on the open-source Xvid codec available at https://www.xvid.com/ and the MPEG-4 file format due to their accessibility and popularity.

2. BACKGROUND INFORMATION

2.1 Motion Compensation

Since this project deals with the compression of MPEG-4 files, it is critical to understand their architecture. MPEG files consist of three major frame types, which are called I frames, P frames, and B frames [6]. These frames minimize the space required to store a video file through a process known as motion compensation. I frames, often referred to as independent frames, contain all the necessary data to display that particular image on the screen. They do not rely on information from any previous frames to be displayed. P frames, by contrast, take advantage of the fact that a frame in a video is likely to be similar to previous frames [6]. Since the colors and general locations of images are likely to be resemble previous frames, P frames use motion vectors to represent the movement of blocks of pixels. Instead of storing all of the information representing the color of each pixel, P frames are able to represent much of the information in the frame as positional changes from the previous frame. B frames are similar to P frames except that they use bidirectional prediction. This means they not only use motion vectors to predict the motion of future blocks of pixels, but they also use backward prediction to represent the locations of blocks at previous frames. While more predictive frames use less space, it is necessary to use independent frames as a starting off point from which predictions can be made. Thus, MPEG files contain a sequence of I, P, and B frames, as is exemplified in the figure.



2.2 Transform Coding

Another important compression technique MPEG files use is known as transform coding [6]. Transform coding divides an image into 8 x 8 pixel blocks. Instead of storing the color code for each pixel in the image, transform coding uses a discrete cosine transformation to represent the approximate color for each of the pixels. For small size blocks of pixels, the difference isn't noticeable to the human eye, but it largely reduces file size.

2.3 Entropy Encoding

The final technique MPEG files use to compress data is entropy encoding. Entropy encoding is a method of compression that takes advantage of the fact that certain sequences of bits may appear more often than others by representing more common sequences with fewer bits [6]. It replaces longer sequences of bits that appear often in a file with a shorter sequence specifically meant to represent the longer sequence. This reduces the overall file size, since the most common information requires the least space to store.

2.4 Features

One important aspect of compression algorithms for MPEG files is that they must allow a certain set of core features. Compressing a video file is far less useful if it compromises the ability to use basic features one would expect when watching a video. For example, a good compression algo-

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

rithm must allow for random access [3]. From a practical standpoint, users watching a video will often want to skip to a particular part in the video. While minimizing the file size may be helpful when storing or transferring the information, the user still needs to be able to access random parts of the video for the compression algorithim to be practical. Other important features of an MPEG compression algorithim include forward searches, reverse playback, audio-visual syncronization, and robustness to errors [3].

3. PROBLEM DEFINITION

The problem with data compression is that it is nearly always a trade off. A video file, for instance, could be easily reduced to require much less space, but this would require significantly lowering its quality. Conversely, a video that it higher quality may look better, but will take up more space. Data compression algorithms attempt to do a reasonable job of both; they attempt to maintain a decent level of quality while also keeping the file size as small as possible. To address this problem, people are always experimenting with better methods of storing data.

4. PROJECT DESIGN

4.1 Solution

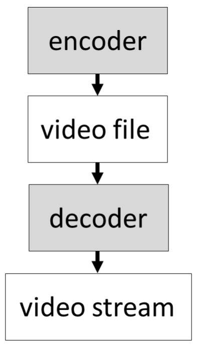
Much of the research thus far in this field is focused on minimizing the storage space of a video file with minimal, if any, compromises made in quality. While this approach may be elegant, it is not always practical. Sometimes slightly larger compromises should to be taken if a video is being watched over a poor internet connection, or the screen being used to watch a video is small. Current implementations of compression algorithms also do not take into consideration where the focus of the viewer is likely to be. A viewer's eyes are likely focused on the center of the screen, causing them not to pick on details in the corners of the screen. Much space is wasted storing this information, even though most users will not notice it. I plan to address this issue in my project. I also intend to experiment with increasing the size of the blocks used for the discrete cosine transformation. The quality difference may not be particularly noticeable, and there is little research using blocks at any size other than 8x8 pixels.

4.2 Hardware

Video files and compression algorithms are commonplace on many modern electronic devices. I can code algorithms and test video quality on hardware already available to me, and thus I will not require any additional hardware for this project.

4.3 Software

The software that I plan to use is called Xvid, which is a free, open source codec for MP4 files. I will use it as a starting point, and will attempt to further its compression capabilities using the ideas previously described, and any new ideas I discover as I experiment with the software. Below is a visual representation of how the encoder, data file, and decoder interact to store and play video.



4.4 Budget

Since the software is free and no new hardware will need to be acquired, the project should not have any expenses.

5. TIMELINE

- October 21: Be familiarized with the Xvid codec, how it works, and how to make simple modifications to it to change compression.
- November 6: Have unique, decently working, personal compression algorithm. At this point I will have explored Xvid and experimented with ideas for some time, so I hope to have added some of my own ideas to the codec.
- November 8: Complete a general outline of the paper to serve as a guide.
- November 16: Complete the first draft of the paper.
- November 30/December 4: Be prepared for project presentation.
- December 12: Finish second draft of paper.
- December 16: Finish final draft of paper and software.

6. REFERENCES

[1] 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.

[2] 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.

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

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

[5] 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.

[6] 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.