CPSC 4660 Final Report

Video Data Compression by Quadratic Bézier Curve Fitting

Chase Mossing, Brendan Ball
Feb 9, 2023

Summary of Implementation

As Khan mentioned in the article, “using quadratic Bézier curve fitting [is] a new and efficient way for video data compression” (Khan 2010). The method estimates the luminance or color variations of a spatial location in a sequence of given frames using quadratic Bézier least square fitting. It then outputs the quadratic Bézier control points and gives the difference between the original and fitted data. The method can then be applied to one dimensional spaces like luminance and chrominance components separately or three dimensional color spaces like RGB. The paper goes on to describe the specific quadratic Bézier curve formula used, and the method of fitting the curves to the raw video data - which are to be implemented in the component. We accomplished this using C++ and opencv to get the information out of the videos that the user uploads. Since the paper gave the necessary formulas and calculations, we just had to implement the formulas and algorithms processes described in the paper, and measure our results against those depicted in the paper.

Implementation details

Using C++ and opencv, we are implementing the mathematical formulas provided by the paper. We then recreated them in the codebase. This produced similar, if not the same results as the one provided in the paper. The program starts out with the user providing a file path to the video that will be used. An opencv Reader is used to read the video data into an array of opencv Mat objects representing the frames of the video. It then splits the video frames into a user specified amount of segments, then for each segment, the color values of each pixel from each frame in the data are extracted into a multidimensional array. Then the program calculates a set of quadratic Bézier curve control points for each color channel, in each pixel, for each segment using a least

square fitting formula (2, Khan, 2010), derived from the quadratic Bézier curve formula (1, Khan, 2010):

(1) $Q(t_i) = (1 - t_i)^2 P_0 + 2t_i (1 - t_i) P_1 + t_i^2 P_2$

(2) $U = m\sum i=1 [O_i - Q(t_i)]^2$

For a given color channel in a given pixel location, the start and end control points will be the values in the first and last frame of that segment. The least square fitting formula is used to calculate a middle control point. These control points are organized as size 3 tuples in a tensor that is shaped according to the segment amount, width, height, and color scale, (seg_num, row, col, color) and written as a binary stream to a file on disk for later decoding.

The decoding starts by reading the control points from the aforementioned file, then interpolating the video data for each frame by using the quadratic Bézier curve formula above (1, Khan, 2010). This video data is then written to an mp4 file using an opencv Writer.

Evaluation strategies

We employed three evaluation strategies with this project. The main comparison we wanted to make was the difference between the raw video data size after being read into memory, and the size of the calculated control point data. The other two secondary evaluation strategies we had were comparing the video length between the original and compressed data, and the quality difference between the original video, and the newly created control point video. With that being said, we wanted to ensure that the process and implementation we were using was close to the results of the paper.

Results of the evaluation

After the program was fully finished, the testing phase was started. That gave us an opportunity to download a royalty free, and public domain video that we used for testing purposes. Starting off with file size, we found that the file size did in fact end up being smaller than the original. The raw data from the original video used, was around 3GB once it was loaded into memory, and the control point data used for the creation of the new file, was around 500MB. This means that the newly created control points data is about ~83-84% smaller than the original video data. To address the second point we have introduced, we wanted to compare video length between the original video, and the newly compressed video. We found the video length of the original video file and the newly created file to be equal to the same length as expected. As for the third strategy mentioned, we wanted to address the difference in quality between the original and newly compressed and fitted data.

From our implementation of the proposed method from the paper, we saw a lot of visual differences between the original video and the newly compressed video. There were signs of ghosting and stuttering. There were suspicions on our end that it could have been opencv's library that was causing the issue with the data structures we were using. However, we could not prove that that was the issue that was causing our problems. After extensive refactoring and debugging we had no luck at fixing this issue. While the first two evaluation strategies still stand correct, this last one was where we started to see issues in our final result. While this is most likely an issue on our end with the implementation that is not to say more time, or different approaches could have fixed the issues.

Conclusions drawn from the experience

After completing this experience, we found that the paper was quite easy to read, and included helpful information on how to implement this in any language necessary. The paper was well written for the most part, and provided more than enough information to implement a working version of video data compression using quadratic Bézier curves. As we did not recreate other versions of compression, we did not want to include comparison information for the other compression methods.

One issue we ran into made the final compressed video almost complete, but had ghosting and stuttering. The evaluation showed a significant reduction in file size, with the newly created control point data being about 83-84% smaller than the original video data. Also, the video length of the original video file and the newly created file was found to be equal. However, our implementation was not fully successful in recreating the exact video from start to finish. We ran into one issue that seemed to make extreme flaws in the final result that was being outputted. While we were close to finishing we could not quite iron out these issues we ran into on time, but it could be a potential avenue for further evaluation and implementation. This does not mean that the paper is incorrect by any means, but in the allotted time frame that was given, we were unable to recreate a similar version to the paper that we were following.

Overall, this method of compression provides an alternative approach to video compression, and it has the potential to be useful in situations where storage space is limited. It is also versatile, as it can be applied to both one-dimensional and three-dimensional color spaces, and it can be used

for both luminance and chrominance components separately. The method's ability to approximate input points using quadratic Bézier least square fitting makes it one of many efficient options towards compression.

Citations & References

Khan, M.A. (2010). A new method for video data compression by quadratic Bézier curve fitting.

Signal, Image and Video Processing, 6(1), 19-24. DOI: 10.1007/s11760-010-0165-9.