CAV - Developing a Video Codec

Nuno Humberto Nuno Miguel Silva

***Abstract* – One of the most used approaches to losslessly compress data takes advantage on the usage of Golomb coding. In lossless video compression, it is used for encoding image prediction residuals.**

The main objective of this assignment is to develop a lossless video codec that produces a compressed file as small as possible. By taking advantage of intra-frame and inter-frame coding, the final compressed size can be successfully minimised.
***Keywords* – video compression, lossless codec, JPEG-LS**

I Scenario

I-A *Description*

In this project, just a first stage of the lossless video codec was developed. This report will describe the process of both encoding and decoding techniques.

The samples used for this were found in the *eLearning* page, in a provided link to a website with video samples.

II Encoding

II-A *Stage I - Frame Acquisition*

The first stage of performing an encoding operation is the frame loading stage - it relies on acquiring a frame from the input video file.

For each chroma subsampling type there is a different way of accessing the respective YUV values. Here are the planar storing formats:

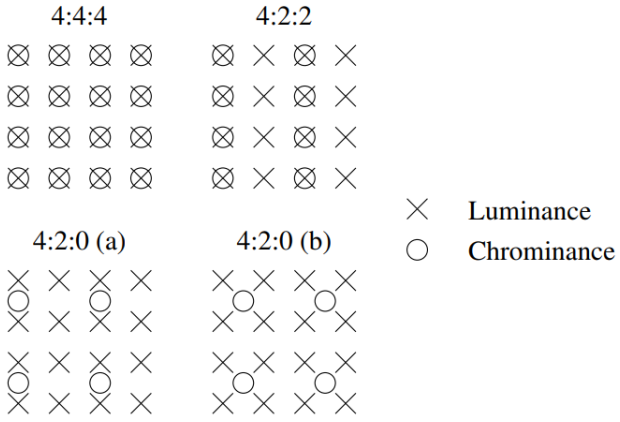

Figure 1: Chrominance subsampling

Having this knowledge, we can access the planar stred values fo the YUV triplet using the following deduced equations:

$$\begin{cases} y = imgData[i] \\ u = imgData[i + (yRows \cdot yCols)] \\ v = imgData[i + (yRows \cdot yCols) * 2] \end{cases}$$

Figure 2: YUV storage equations for 4:4:4 format

$$\begin{cases} y = imgData[i] \\ u = imgData[i + (yRows \cdot yCols)] \\ v = imgData[\frac{i}{2} + (yRows \cdot yCols) \cdot \frac{3}{2}] \end{cases}$$

Figure 3: YUV storage equations for 4:2:2 format

$$\begin{cases} y = imgData[i] \\ nRow = \frac{\frac{i}{yCols}}{2} \\ a = \frac{i}{2} \mod yCols + \frac{nRow - (nRow\%2)}{2}) \cdot yCols \\ size = (yRows \cdot yCols) \\ u = imgData[a + size] \\ v = imgData[a + size \cdot \frac{5}{4}] \end{cases}$$

Figure 4: YUV storage equations for 4:2:0 format

This is made possible because it is known that the U values come after the Y, and the V values come after the U.

II-B *Stage II - Predictor Calculation*

The second stage of the encoding procedure is the predictor calculation. In our encoder, the predicted values are obtained using the nonlinear predictor of JPEG-LS. It uses a predictor based on the same spatial configuration as that of JPEG. However, instead of a linear predictor, it uses the nonlinear predictor:

$$\hat{x} = \begin{cases} \min(a, b) & \text{if} \quad c \geq \max(a, b) \\ \max(a, b) & \text{if} \quad c \leq \min(a, b) \\ a + b - c & \text{otherwise} \end{cases}$$

Figure 5: JPEG-LS non-linear predictor

We use this equation in order to obtain the predicted values used further to calculate the residuals. The values from the first row and the first column are copied from the original sequence in order for the rest to be generated with the previous equation.

II-C Stage III - Residual Calculation

The third stage has the function of calculating the residuals. This operation is achieved by taking the amounts of the original sequence and subtracting the values of the predicted sequence. This procedure is successfully achieved by using the following equation:

$$residual_i = original_i - predicted_i$$

II-D Stage IV - Golomb Coding

Firstly, all residuals have to be non-negative, and for this the numbers closest to zero have to remain the closest possible to it when passing from negative to positive. The following equation is used for this first step:

$$\begin{cases} target = x \times 2, \text{ if x is positive} \\ target = -x \times 2 - 1, \text{ if x is negative} \end{cases}$$

Two parameters are crucial when writing the Golomb encoded residuals to the file. An m value is going to be used for encoding the file. The encoder then proceeds to calculate the q and r for each target residual:

$$\begin{cases} q = \lfloor \frac{n}{m} \rfloor \\ r = n - qm \end{cases}$$

II-E Stage V - Writing to a file

The previously calculated quotients and remainders are the written to a file.
The following table depicts the obtained results.

Table I: File size comparison

Symbol Space Division	Original Size (KB)	Final size (KB)
akiyo_cif.y4m	45621	51477
football_422_cif.y4m	72992	84454

III Decoding

III-A Stage I - Recovering the original values

The objective of the first part in the decoder is to recover the predicted values. In this step, the residuals are used and the values of the first column and first row are copied to the predicted matrix. Then, the JPEG-LS predictor used earlier in the encoding process is re-used in this phase to recover the rest of the predicted values. The original values are retrieved by adding the predicted values to the residuals. The following equation show how this can be possible:

$$original_i = predicted_i + residual_i$$

IV Conclusion

Partial objectives were achieved successfully.

Although some curious insights were acquired during the development of this assignment, the main objectives were not achieved. It was interesting to observe that the usage of the JPEG-LS predictor did not provide smaller dimensions than the original video samples that were tested. We suspect that the usage of the LOCO-I predictor will not yield good compression results until the usage of context modeling is implemented.

$$g_1 = D - B$$
$$g_2 = B - C$$
$$g_3 = C - A$$

Figure 6: Context modeling

The main issues in the development process of this project were based on finding strategies to work around the problems we had. In general, and how these problems were resolved, we have learned something from these somewhat unfamiliar obstacles although this assignment is incomplete.

References

https://jpeg.org/jpegls/index.html
http://www.labs.hp.com/research/info_theory/loco/