Universidade de Aveiro(UA)

DETI

Information and Coding

Project 3

Eduardo Lopes Fernandes,
João Afonso Ferreira,
Rafael Luís Curado

Teacher: António Neves
Group: P1 - G1

October 2023

Contents

1 Part I **1**
1.1 Image 1
1.2 Video 2
1.3 Encoder 2
1.4 Decoder 3
1.5 Results 4

2 Part III **5**
2.1 Encoder 5
2.2 Decoder 6
2.3 Results 7

3 Acronyms **8**

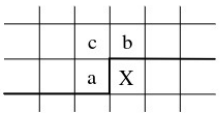
4 Bibliography **9**

List of Figures

1 The seven linear predictors of lossless JPEG. 1
2 Image Codec Results 4
3 Video Codec Results 4
4 Lossy Video Codec Result 7

1 Part I

Utilizing grayscale simplifies the problem by reducing the image to a single channel. The subsequent step involves computing a variety of predictors, specifically employing the seven linear predictors inspired by JPEG.



Mode	Predictor
1	a
2	b
3	c
4	$a + b - c$
5	$a + (b - c)/2$
6	$b + (a - c)/2$
7	$(a + b)/2$

Figure 1: The seven linear predictors of lossless JPEG.

In the predictor calculation process, we also derive an essential value known as 'm,' which plays a pivotal role in selecting the appropriate predictor for each image or frame in the case of videos. A lower 'm' signifies enhanced compression efficiency.

The encoded file encompasses a header-like structure (not coded, utilizing a predetermined number of bits) containing crucial information for decoding the image. The header stores values such as 'm' and an identifier representing the selected predictor.

Taking all the aforementioned considerations into account, we can develop a program capable of encoding and decoding images. Extending this functionality to process videos is a straightforward task, involving iteration over all frames.

For coded video files, a primary header contains information such as width, height, and the frame count, followed by a stream of frames. Each frame is accompanied by its header, encompassing the previously mentioned 'm' value and predictor identifier.

1.1 Image

To evaluate the image encoding process, we used three distinct PPM images. The selection of these images was deliberate, aimed at eliciting diverse outcomes. Specifically, we anticipated that the house image, characterized by a high level of color consistency, would undergo the most effective compression. Conversely, we expected the aerial image, with its inherent color variation, to exhibit comparatively less compression. This reasoning stems

from the Golomb coding's proficiency in handling repeated values, making it well suited to capitalize on the color uniformity found in the house image.

.ppm	Size	auto, %	type 1, %	RunTime
arial	1100kb	284kb, 74%	284kb, 74%	0.37s
baboon	768kb	209kb, 72%	209kb, 72%	0.29s
house	193kb	37kb, 80%	40kb, 79%	0.21s

Table 1: Avaliations of three test samples

Table 1 illustrates a compression rate of approximately 75% for each image. However, it's crucial to consider that the original size encompasses all three color channels, whereas the compressed version retains only a single channel.

Our earlier prediction holds true—the house image exhibits the most significant compression. While the difference may seem subtle when analyzing individual images, the impact becomes more pronounced when extrapolating to a video composed of numerous frames. The variation in luminescence content significantly influences the overall compression rate output.

1.2 Video

1.3 Encoder

The **PPM Image Encoder** efficiently compresses grayscale images, extracting luminance from the **Blue, Green, Red (BGR)** color space. The output includes Golomb codes and predictor information for accurate decoding. The code_gray_image function processes the input image, calculates **spatial predictors**, and determines prediction errors. The histograms of errors are used to find the parameter '**m**' for efficient encoding. The main function reads the input PPM image, converts it to grayscale, and encodes the image using Golomb coding. The resulting Golomb parameters and encoded errors are written to the BitStream class developed on previous projects. This codec forms the **basis** for the video codec.

The **Y4M Video Encoder** implemented, extends the principles of the image codec to efficiently compress grayscale video sequences. It uses Golomb coding with adjustable parameters and several predictors in **YUV** (the **Y component** represents the **luminance** of the image, and the U and V components represent the color information) colour space for effective compression. The output includes codes and predictor information, ensuring accurate decoding. The codec processes uncompressed video in the Y4M format, providing a solution for grayscale video compression.

The produced file by the image and video encoders contains a header with essential information (height, width, Golomb parameters), followed by Golomb prediction errors representing the pixel values.The file (in this case called "out_coded_") efficiently represents image or video data in a compact form, allowing reconstruction of the original content.

```
Usage: bin/ppm_code <in_ppm_path> <out_coded_ppm_path>
Usage: bin/y4m_intra_code <in_y4m_path> <out_coded_y4m_path>
```

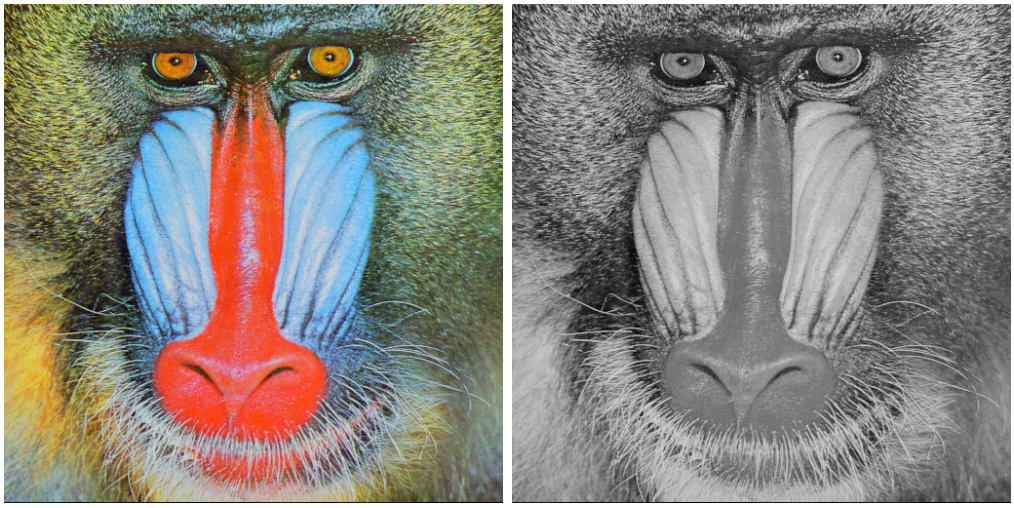
1.4 Decoder

The **Image Decoder** reads the encoded file produced by the respective encoder. Next, it decodes these errors, **reconstructing** pixel values using the specified predictor. The grayscale picture is decoded and then converted to BGR format. The resulting image is saved to an output file, provided by the user. This decoder efficiently reconstructs the original image from the encoded data produced by the corresponding encoder.

Alike the Image Decoder, the **Y4M Video Decoder** reads an encoded file. It then decodes these errors, reconstructing pixel values using the specified predictor. The decoded grayscale frames are written to an output file in Y4M format, which includes the frame headers and pixel values. The resulting Y4M video file can be used to reconstruct the original video sequence from the encoded data produced by the corresponding encoder. The decoder efficiently processes each frame, producing a valid Y4M video file.

```
Usage: bin/ppm_decode <in_y4m_path> <out_coded_y4m_path>
Usage: bin/y4m_intra_decode <in_y4m_path> <out_coded_y4m_path>
```

1.5 Results



(a) Original Image chosen for encoding (b) Grayscaled Image after decoding

Figure 2: Image Codec Results

(a) Original Video chosen for encoding (b) Grayscaled Video after decoding

Figure 3: Video Codec Results

2 Part III

This exercise involves implementing a lossy video codec with a focus on compressing the prediction residuals. The approach includes quantizing these residuals before encoding, introducing a controlled loss of information for compression efficiency. Additionally, a video comparison program "video_cmp" was developed to assess the quality of compressed video sequences. The Peak Signal to Noise Ratio (PSNR) calculation (calculated using the formula below 2) will be used to measure the fidelity of the compressed content compared to the original, as well as the comprehension between compression efficiency and perceptual video quality.

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}} \right)$$

- MAX/A is the maximum possible pixel value (for example, 255 for 8-bit images).
- MSE/e^2 is the Mean Squared Error, calculated as the average of the squared pixel-wise differences between the original and reconstructed images.

$$\text{MSE} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} (I(i,j) - K(i,j))^2$$

  - Where N and M denote, respectively, the number of rows and columns of the video frames.

2.1 Encoder

For this exercise, a **Lossy Video Encoder** was implemented and designed with the introduction of **quantization** for compression. The encoder introduces a quantization factor, denoted as "quantization_factor", which is passed as a command line argument to the program. This factor controls the level of quantization applied to prediction residuals during the encoding process. An essential aspect of this encoder is the calculation of the PSNR after encoding each frame. The encoder reconstructs the frame by adding the quantized residuals back to the predicted values. The quantization step controls the amount of compression that is applied to the video data and has a considerable effect on the quality of the video output. During the quantization step, the encoder divides the coefficients obtained from the transformation step by a set of quantization values [1]. Following the encoding process, the frame is reconstructed by adding the quantized residuals

to the predicted values. Next, the PSNR is determined by comparing the original YUV frame with the reconstructed grayscale frame.

Additionally, after encoding each frame, the PSNR value is printed to the console. This provides valuable feedback and debugging information on the quality of the reconstructed frame compared to the original, helping to evaluate the performance of the lossy compression. Overall, this encoder aims to balance compression efficiency and video quality by incorporating quantization and utilizing the PSNR. The produced file by the image and video encoders contains a header with essential information (height, width, Golomb parameters and the quantization factor).The file (in this case called "out_coded_").

```
Usage: bin/video_cmp <in_y4m_path> <out_coded_> <quantization_factor>
```

2.2 Decoder

It was developed a **Lossy Video Decoder** for this part. When executed with the encoded video file as input, the decoder reads the parameters from the encoded file's header. These parameters include the video's height, width, frame count, parameter "m", prediction method "predictor_t", and the quantization factor "quantization_factor". Leveraging this information, the decoder proceeds to reconstruct each frame. Within the decoding process, the decoding process is going to extract the prediction residuals, which were previously encoded during the video compression. After this, the decoder applies the inverse quantization by multiplying these residuals with the specified read quantization factor from the file. The reconstructed pixel values are then combined with the predicted values, reconstructing the original frame.

The resulting frames, reconstructed through the decoding process, are then written to the output Y4M file.

```
Usage: bin/video_cmp_decode <out_coded_path> <out_y4m_path>
```

2.3 Results

Figure 4: Lossy Video Codec Result

Concluding, the use of quantization in a video codec impacts the relation between compression efficiency and video quality. Adjusting the quantization factor allows for a flexible balance between achieving compression gains and maintaining an acceptable level of perceptual video quality as it is possible to observe on Fig. 4.

3 Acronyms

BGR Blue, Green, Red

PSNR Peak Signal to Noise Ratio

4 Bibliography

References

[1] Lumenci, *Video codecs*, `https://www.lumenci.com/post/how-video-codecs-works`, Accessed: [01/2024], 2023.