Codificação de Áudio e Vídeo (2020/2021)
Deliverable 4 - Video Coding

Agostinho Pires
DETI (of University of Aveiro)
Aveiro, Portugal
(73957) pagostinho@ua.pt

Bruno Pereira
DETI (of University of Aveiro)
Aveiro, Portugal
(84729) babp@ua.pt

Inês Justo
DETI (of University of Aveiro)
Aveiro, Portugal
(84804) inesjusto@ua.pt

Abstract

Neste relatório, fazemos uma breve descrição de cada um dos modos de funcionamento do codec de vídeo desenvolvido. Para cada um dos modos de funcionamento apresentamos resultados de compressão para 4 vídeos no formato YUV. Escolhemos com base nestes resultados a configuração com maior taxa de compressão. Finalmente, na última secção descrevemos um manual de utilização do codec.

1. Codificação sem perdas (Lossless)

Uma codificação ”lossless” tem como objectivo principal a compressão sem existência de perda de informação, nem se quer um bit. Para tal, o uso de métodos preditivos é uma boa abordagem para fazer codificação sem perdas.

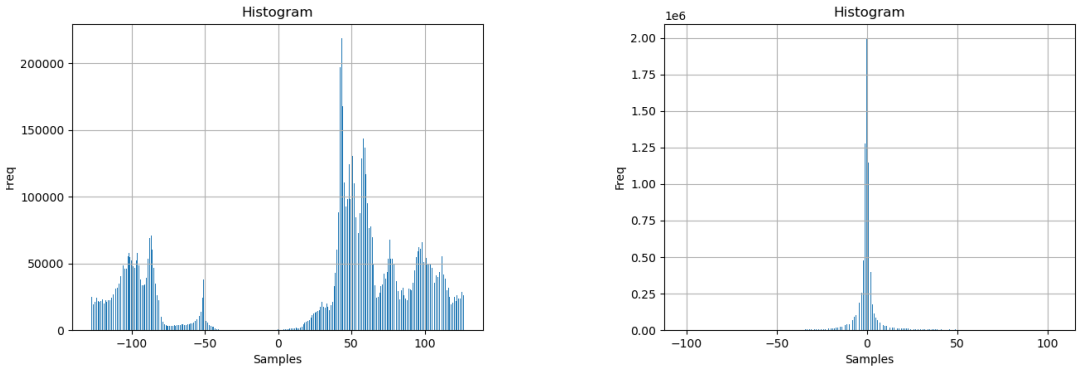
1.1. Modo: intra-frame

O codificador intra-frame implementado, usa métodos preditivos semelhante aos do JPEG, que tem como base os pixeis vizinhos para prever o valor do pixel a codificar, através de um calculo ou relação entre eles. O calculo do residual é efectuado através da diferença do valor do pixel e a previsão, sendo o residual codificado com códigos de Golomb. A descodificação dos residuais é o processo reverso, ou seja, a soma do valor previsto e o residual, de modo a obter o valor original do pixel. Este processo é efectuado para os 3 canais do vídeo no formato YUV.

Mode	JPEG predictor
1	a
2	b
3	c
4	$a + b - c$
5	$a + (b - c)/2$
6	$b + (a - c)/2$
7	$(a + b)/2$
8 (JLS)	$\begin{cases} min(a,b), & \text{if } c \geq max(a,b) \\ max(a,b), & \text{if } c \leq min(a,b) \\ a + b - c, & \text{otherwise} \end{cases}$

TABLE 1: Preditores disponíveis no modo intra-frame.

As próximas figuras são dois histogramas que demonstram o resultado de usar os métodos preditivos em pixeis do canal Y de um video:



(a) Amostras originais

(b) Residuais calculados

Figure 1: Histogramas

O histograma dos residuais segue uma distribuição normal com baixa variância. Como resultado, a entropia desta nova distribuição é menor que a dos valores originais, o que permite uma taxa de compressão superior ao codificar estes residuais usando códigos de comprimento variável, como os de Golomb. Para usar um codificador de Golomb, primeiro mapeamos os residuais (que podem ser ambos inteiros positivos e negativos) para apenas inteiros positivos:

$$Mapped = \begin{cases} 2 * residual, & \text{if } residual \geq 0 \\ -residual - 1, & \text{if } residual < 0 \end{cases}$$

Isto resulta na seguinte distribuição geométrica (Figure 2):

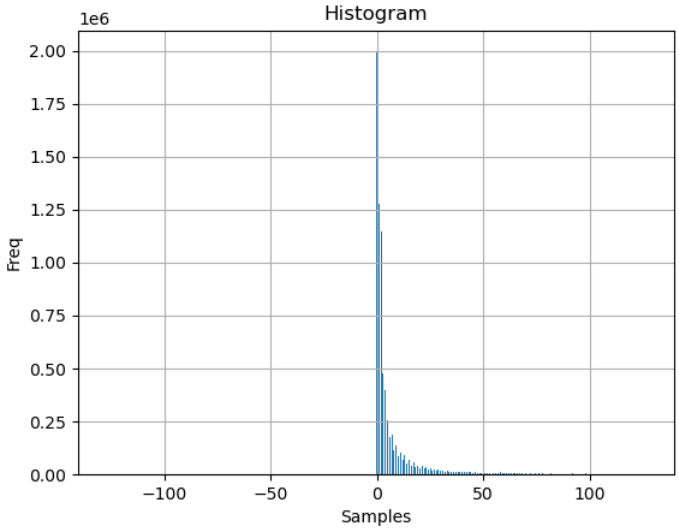


Figure 2: Residuais mapeados para valores positivos

Como os códigos de Golomb são ótimos para distribuições geométricas, segui-se uma estratégia de adaptação do parâmetro m à média dos últimos n residuais mapeados, de forma a aproximar a média real da distribuição geométrica e obter códigos mais curtos.

1.2. Resultados: intra-frame

Nesta secção estão apresentados resultados de compressão do modo intra-frame nos 4 vídeos recomendados no enunciado deste projeto. Os 4 vídeos têm todos 660MB de tamanho, resolução 1280x720, 4:2:0 chroma subsampling, 50 fps e 10 segundos de duração. Estes resultados estão expresso na forma de rácio de compressão e de bitrate em bits/segundo (bits por pixel por segundo). Em cada teste variamos o preditor (usámos 5 dos 8 disponíveis). Para cada vídeo descodificamos apenas um teste para demonstrar o tempo de descodificação.

$$CompressionRatio = \frac{originalSize}{compressedSize}$$

$$Bitrate = bitsMédiosPorPixel * framesPorSegundo$$

$$bitsMédiosPorPixel = \frac{númeroTotalBits}{númeroTotalPixeis}$$

JPEG predictor	Encoding time	Decoding time	Compressed size (MB)	Compression ratio	Bitrate (bps)
1	34m17.495s	18m33.266s	401	1.645885287	364.9991111
5	34m19.478s	-	403	1.637717122	366.8195556
6	31m52.611s	-	447	1.476510067	406.8693333
7	37m27.976s	-	435	1.517241379	395.9466667
8	35m17.718s	-	405	1.62962963	368.64

TABLE 2: Video "ducks_take_off_420_720p50.y4m" 660MB.

JPEG predictor	Encoding time	Decoding time	Compressed size (MB)	Compression ratio	Bitrate (bps)
1	34m28.649s	17m16.666s	400	1.65	364.0888889
5	33m55.783s	-	389	1.696658098	354.0764444
6	29m45.482s	-	390	1.692307692	354.9866667
7	35m18.872s	-	379	1.741424802	344.9742222
8	35m6.709s	-	381	1.732283465	346.7946667

TABLE 3: Video "in_to_tree_420_720p50.y4m" 660MB.

JPEG predictor	Encoding time	Decoding time	Compressed size (MB)	Compression ratio	Bitrate (bps)
1	34m23.390s	17m11.771s	380	1.736842105	345.8844444
5	34m26.248s	-	369	1.788617886	335.872
6	29m2.930s	-	372	1.774193548	335.872
7	34m50.341s	-	365	1.808219178	332.2311111
8	34m29.751s	-	358	1.843575419	325.8595556

TABLE 4: Video "old_town_cross_420_720p50.y4m" 660MB.

JPEG predictor	Encoding time	Decoding time	Compressed size (MB)	Compression ratio	Bitrate (bps)
1	34m40.118s	17m7.673s	460	1.434782609	418.7022222
5	34m28.741s	-	449	1.469933185	408.6897778
6	31m41.817s	-	452	1.460176991	411.4204444
7	37m2.157s	-	442	1.49321267	402.3182222
8	34m26.121s	-	443	1.489841986	403.2284444

TABLE 5: Video "park_joy_420_720p50.y4m" 660MB.

Comando usado para obter o melhor resultado (no video "old_town_cross_420_720p50.y4m"):

```
./codec encode old_town_cross_420_720p50.y4m result 8 0
```

1.3. Modo: hybrid

O codificador hybrid implementando usa uma combinação de compressão intra-frame (como descrita anteriormente) com compressão inter-frame. A compressão inter-frame usa o método preditivo de compensação de movimento, começando por efectuar uma divisão do frame a codificar em blocos, comparando com os blocos do frame anterior, os que tenham menor diferença em posição diferente é considerando movimento, a partir da posição desses blocos é calculando o vector de movimento seguido dos residuais das diferenças, que são codificados novamente através de códigos de Golomb. Sendo a descodificação a inversão do processo, muito semelhante ao processo inter-frame, descrito anteriormente. A compressão intra-frame é usada num único frame a uma taxa pré-definida.

1.4. Resultados: hybrid

Não foi terminada a implementação do modo de funcionamento hybrid pelo que não foi apresentado resultados.

2. Codificação com perdas (Lossy)

2.1. Modo Sequencial (Baseline) do JPEG

Para uma codificação com perdas, teve-se como base a quantização do modo sequencial do JPEG, com algumas modificações.

Primeiramente, é calculada a matriz DCT:

- os residuais da imagem do video a ser codificada, são agrupados em blocos/matrizes de 8x8 (B).
- A cada valor do pixel na matrix B é subtraído o valor 128.
- É calculada a matriz DCT: $D = TBT'$, em que D é a matriz quantizada, T é a matriz de transformação da DCT, T' a transposta de T e B a matriz original.

De seguida, os coeficientes da DCT são quantizados:

- Os coeficientes são quantizados usando uma abordagem de Threshold coding: $D(r,c) = ROUND(D(r,c)/Q(r,c))$, onde $D(r,c)$ corresponde a cada elemento da matriz D e $Q(r,c)$ a cada elemento da matriz de quantização. A matriz de quantização Q usada corresponde à matriz quantização do JPEG da componente a ser quantizada (Fig. 3).

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Figure 3: Matrizes de quantização do JPEG. A primeira é relativa à componente da luminância e a segunda às componentes da crominância.

- Finalmente, os coeficientes são codificados. Para o valor do coeficiente DC, codifica-se a diferença do anterior. Nesta implementação, os coeficientes de uma imagem são codificados através do código de Huffman, e a àrvore de Huffman calculada para a imagem é codificada através do código de Golomb (Fig. 5). Primeiro é codificada a árvore e de seguida o código de Huffman. De forma a separar cada código de Huffman e Golomb da árvore da imagem seguinte, utilizámos um terminador para o código de Huffman (-3). Como já se possuía um terminador do último bloco, mudou-se a abordagem mais comum de codificação de um par EOB para um par BOB (beginning of block) que contivesse um -1 no valor que representa o número de zeros precedentes e o valor do coeficiente de DC no valor que representa o coeficiente, como se pode verificar na Fig. 4.
- Na construção da árvore de Huffman, de forma a usufruir do facto de usar codificação run length dos coeficientes, em forma de pares de valores, calcula as probabilidades do primeiro valor de cada par e as do segundo valor de cada par em separado, para que pudessem ter o mesmo código associado. Assim, na árvore, à excepção de casos como o valor de terminação da árvore, o valor de começo do bloco, e o valor da diferença de coeficientes DC,

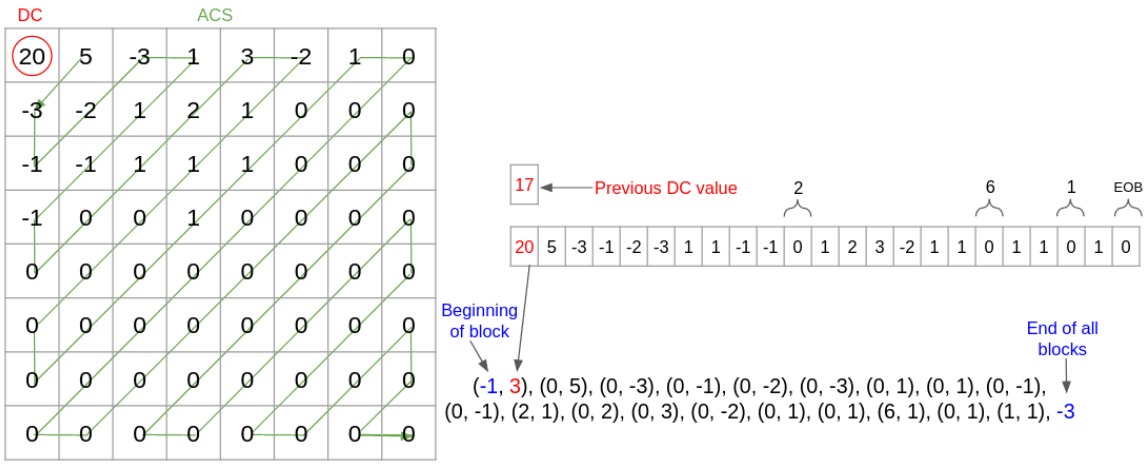


Figure 4: Codificação Run-length dos coeficientes.

os valores que representam o número de coeficientes AC iguais a zero encontram-se sempre em folhas à esquerda e os valores que representam coeficientes AC diferentes de zero nas folhas à direita (Fig. 5).

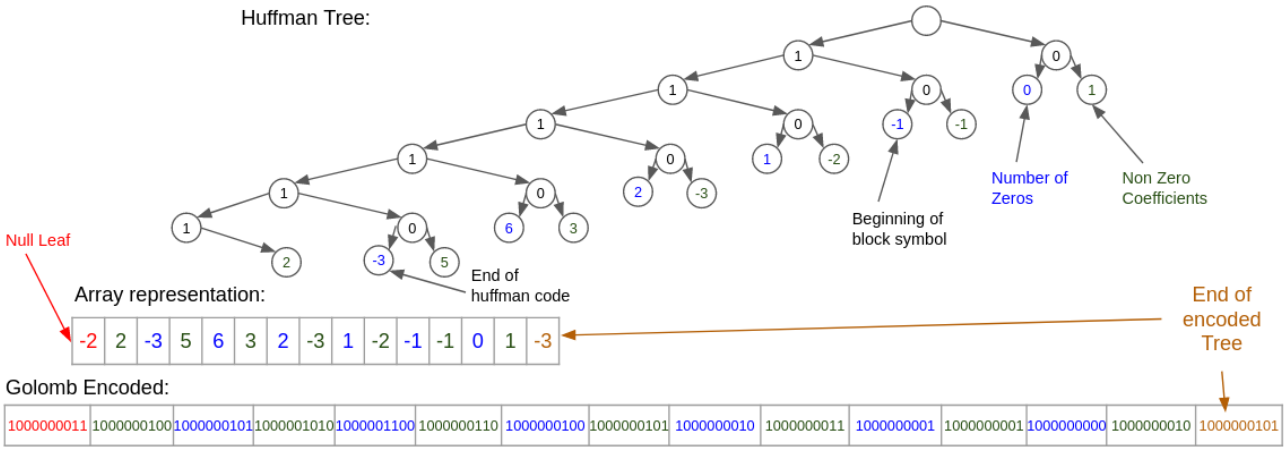


Figure 5: Construção e codificação da Huffman Tree.

Infelizmente, não conseguimos deixar funcional o processo de colocar os valores descodificados na matriz do descodificador.

3. Extra mile

Com a utilização da biblioteca ffmpeg, pode-se efectuar manipulação de ficheiros, alterar os formatos de vídeo (MP4, AVI, MOV) e áudio (WAV, MP3). O codificador implementado faz tratamento de video no formato .avi ou .mp4, extraindo o áudio do vídeo e codifica por separado em diferentes ficheiros. Usa um método intra-frame com o preditor JPEG-LS sem perdas, para o vídeo e um método preditor polinomial de ordem 3, para o áudio. A descodificação é o processo reverso ao implementado na codificação, descodificando o áudio e o vídeo, sendo o áudio sempre recuperado no formato .wav, usa a ffmpeg para juntar novamente o áudio e o vídeo num ficheiro de vídeo reconstruído.

4. Manual

As funcionalidades do codec estão divididas em dois comandos: encode e decode.

```
usage: ./codec <operation>
operations:

encode  encode SOURCE .y4m video file into DEST compressed file
        usage:
                encode SOURCE DEST <predictor [1,8]> <mode 0 - intra, 1 - hybrid> [options]
        options:
                -hist   compute histograms and entropy and save to .csv
                -lossy  use lossy compression
decode  decode SOURCE compressed file into DEST .y4m video file
        usage:
                decode SOURCE DEST
```

Exemplo para codificar um ficheiro ”video.y4m” sem perdas usando o preditor de JPEG 5 e o modo de funcionamento intra frame, e guardar o resultado comprimido num novo ficheiro ”test”:

```
./codec encode video.y4m test 5 0
```

O exemplo anterior mas com perdas:

```
./codec encode video.y4m test 5 0 -lossy
```

Exemplo para descodificar um ficheiro comprimido ”test” e guardar o resultado descodificado num novo ficheiro ”test.y4m”:

```
./codec decode test test.y4m
```

Tal como codec, o extra (extra mile) também se encontra dividido em dois comandos de encode e decode.

```
usage: ./extra <operation>

operations:
encode ->  <source> <auido code> <video code>

decode ->  <audio code> <video code> <destiny>
```

Exemplos:
encode:

```
./extra encode ../videos/input5s.avi code_audio code_video
```

decode:

```
./extra decode code_audio code_video ../videos/new.avi
```