UNIVERSIDADE FEDERAL DO PAMPA

Luiz Daniel Garay Trindade

Análise do processo de codificação aritmética binária adaptativa ao contexto dos padrões H.264 e HEVC

Alegrete
2018

Luiz Daniel Garay Trindade

Análise do processo de codificação aritmética binária adaptativa ao contexto dos padrões H.264 e HEVC

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Bruno Boessio Vizzotto

Alegrete
2018

Luiz Daniel Garay Trindade

Análise do processo de codificação aritmética binária adaptativa ao contexto dos padrões H.264 e HEVC

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Trabalho de Conclusão de Curso defendido e aprovado em 07 de DEZEMBRO de 2013
Banca examinadora:

Bruno Boessio Vizzotto

Prof. Dr. Bruno Boessio Vizzotto
Orientador
UNIPAMPA

Crístian Müller

Prof. Me. Crístian Müller
UNIPAMPA

Eliezer S. C Flores

Prof. Me. Eliezer Soares Flores
UNIPAMPA

Resumo

Aplicações que utilizam vídeos digitais estão cada vez mais presentes em nosso cotidiano. A sua utilização vem aumentando por conta do seu fácil acesso, bem como pelo aumento da qualidade, podendo ser observado em smartphones, smart TVs, video games, entre outros dispositivos que suportam codificação e decodificação de vídeo digital. Para que seja possível o processamento, armazenamento e transmissão dos vídeos, é necessário que seja feita a compressão dos dados que os representam. O HEVC é o padrão estado-da-arte de codificação de vídeo, ele foi desenvolvido com o objetivo de dobrar a eficiência de codificação em relação a seu antecessor H.264. Um compressor de vídeo possui vários módulos, que em conjunto realizam a codificação de vídeo. Entre esses módulos destaca-se a codificação de entropia responsável pela geração do *bitstream*, saída do codificador. A codificação de entropia do padrão HEVC é composta pela codificação aritmética adaptativa ao contexto (CABAC), enquanto o H.264 contém adicionalmente a codificação de comprimento variável adaptativa ao contexto (CAVLC). Este trabalho apresenta uma revisão bibliográfica dos codificadores de entropia dos padrões HEVC e H.264, ademais, apresenta trabalhos relacionados a proposta de análise do comportamento deste módulo, utilizando os softwares de referência dos respectivos padrões. Estas avaliações têm por objetivo verificar a possibilidade da utilização de técnicas de agrupamento de dados para possibilitar paralelismo na codificação de entropia.

Palavras-chave: Codificação de Vídeo. HEVC, Codificação Aritmética, CABAC.

Abstract

Applications that use digital videos are increasingly present in our daily lives, their use has been increasing due to the easy access, and also by the increase of quality, which can be experienced in smartphones, smart TVs, video games and other supported devices. However, to be possible the processing, storage and streaming of video content, it is necessary to compress the data that represent them. The HEVC is the state-of-the-art video coding standarda and it was developed with the goal of doubling the coding efficiency over its predecessor H.264. The video encoder have several modules. Among these modules there is the entropy encoder, responsible for the bitstream generation. The HEVC entropy coding standard consists in the context-adaptive binary arithmetic coding (CABAC), while H.264 has also the context-adaptive variable length coding (CAVLC). This work presents a extended review of the entropy encoders in the HEVC and H.264 standards. Moreover, presents related works and evaluation of the behavior of this module, using the reference software of the respective standards. These evaluations verifies the possibility of using parallel techniques in the entropy coding of both standards.

Key-words: Video Encoding. HEVC. Arithmetic Coding. CABAC.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

2D Duas Dimensões

AVC *Advanced Video Coding*

CABAC *Context-Based Adaptive Binary Arithmetic Coding*

CAVLC *Context-Based Adaptive Variable Length Coding*

CMY *Cyan, Magenta, Yellow*

CMYK *Cyan, Magenta, Yellow, Black*

CTU *Coding Tree Unit*

CU *Coding Unit*

DCT *Discrete Cosine Transform*

DST *Discrete Sine Transform*

EXPG *Exponential-Golombcoding*

FDCT *Forward Discrete Cosine Transform*

FL *Fixed length*

GOP *Group of Pictures*

HEVC *High Efficiency Video Coding*

HM *HEVC Test Model*

IEC *International Electrotechnical Commission*

ISO *International Organization for Standardisation*

ITU *International Telecommunication Union*

ITU-T *International Telecommunication Union - Telecommunication*

JCT-VC *Joint Collaborative Team on Video Coding*

JM *Joint Test Model*

LDP *Low Delay P*

LPS *Less Probable symbol*

MC *Motion Compensation*

ME *Motion Estimation*

MPEG *Moving Picture Experts Group*

MPS *Most Probable Symbol*

MV *Motion Vector*

P *Predictive*

PU *Prediction Unit*

RALC *Random Access Low Complexity Profile*

RAP *Random Access Profile*

RGB *Red, Green, Blue*

SAD *Sum of Absolute Differences*

SATD *Sum of Absolute Transformed Differences*

T *Transform*

TU *Transform Unit*

U *unary*

VCEG *Video Coding Experts Group*

YCbCr *Luminance, Chrominance Blue, Chrominance Red*

Sumário

1 Introdução

O uso dos vídeos digitais vem crescendo a cada ano que passa tanto em aplicações voltadas a TV digital quanto em vídeos sobre a internet, sendo nos smartphones ou em outros aparelhos que suportam vídeo digital. Com o aumento na qualidade visual dos vídeos também cresce o tamanho dos dados necessários para representá-los, tendo assim um maior volume de dados dificultando o processamento, armazenamento e também a transmissão. Para resolver esse problema são utilizados compressores de vídeos. Porém, quanto maior o volume de dados e maior a exigência por excelência em qualidade visual, os compressores tornam-se também cada vez mais complexos, desta forma, tanto a academia quanto a indústria investem no desenvolvimento de técnicas de compressão e padrões mais eficientes.

Sem a utilização de compressão de vídeos, o volume de dados para processamento, armazenamento e transmissão seria muito elevado, sendo necessário um grande espaço de armazenamento e também uma grande largura de banda para a transmissão. Por exemplo, para um vídeo com resolução de 720x480 *pixels* a 30 quadros por segundo e com 24 bits por *pixel*, a taxa de transmissão seria aproximadamente 249 milhões de bits por segundo (249Mps). Para o armazenamento de 10 minutos de vídeo seriam utilizados quase 19 bilhões de bytes (19GB). Já para os vídeos que possuem uma resolução maior, por exemplo, os vídeos de 1920x1080 *pixels* a 30 quadros por segundo com 24 bits por *pixel*, a taxa para transmitir é de aproximadamente 1,5 bilhões de bits por segundo (1,5GB) e para o armazenamento de um vídeo com 10 minutos, 112 bilhões de bytes que seriam 112GB (AGOSTINI, 2007).

Um codificador de vídeo tem como objetivo reduzir o volume dos dados que representam um vídeo. Para obter a diminuição dos dados, os compressores reduzem os dados redundantes considerando que as informações retiradas são menos perceptíveis pelo sistema visual humano. O compressor de vídeo possui alguns blocos de codificação na qual cada bloco é responsável por reduzir um tipo de redundância. O bloco de predição intra-quadro é responsável por reduzir a redundância espacial, o bloco de predição inter-quadros reduz a redundância temporal e a codificação de entropia é responsável por reduzir a redundância entrópica.

A codificação de entropia, diferente dos outros módulos do compressor de vídeo, não possui perda e é o último módulo do processo de codificação de vídeo e o primeiro módulo da decodificação. O bloco de codificação de entropia, segundo o padrão H.264/AVC (*Advanced Video Coding* (ITU-T, 2007)), possui dois métodos de codificação. A codificação de comprimento variável adaptativo ao contexto (CAVLC) e a codificação binária aritmética adaptativa ao contexto (CABAC). E no padrão de codificação de vídeo H.265/HEVC (*High Efficiency Video Coding* (ITU-T, 2018)) somente o método CABAC é utilizado para codificação de entropia.

Com o aumento na qualidade dos vídeos e consequentemente no tamanho as técni-

cas e padrões de compressão de vídeo estão sendo aprimorados para poderem comprimir os vídeos visando eficiência e pouco custo de energia. O codificador de vídeo de alta definição HEVC é um dos mais novos padrões de compressão de vídeo sendo o sucessor do padrão H.264 e veio com o objetivo de dobrar a taxa de compressão em relação aos padrões anteriores. O HEVC foi desenvolvido pela equipe colaborativa conjunta em codificação de vídeo (JCT-VC) uma colaboração entre a organização internacional de normalização (ISO/IEC MPEG) e o setor de padronização de telecomunicações (ITU-T VCEG), que faz parte da união internacional de telecomunicações (ITU). Tendo como objetivo dobrar a eficiência de codificação, em comparação com outros padrões, o HEVC possui um nível de complexidade muito maior. O HEVC foi criado por conta da grande popularidade da TV digital de alta definição do acesso dos vídeos por dispositivos móveis e outros aplicativos multimídias.

Este trabalho apresenta uma proposta para avaliação do módulo de codificação de entropia do codificador aritmético binário adaptativo ao contexto dos padrões H.264 e HEVC. Com essa avaliação foi possível obter dados estatísticos acerca do comportamento dos elementos sintáticos no processo de codificação de vídeo digitais. Para que as avaliações fossem possíveis, foram feitas alterações nos arquivos de configuração e nos softwares de referência dos respectivos padrões na qual foram incluídos códigos próprios para simplificação do processo de codificação de elementos sintáticos no processo de codificação de entropia. Com os dados obtidos, foi feita uma compilação para relacionar a geração de dados de elementos sintáticos dependentes e independentes observando seu comportamento dinâmico. Após as avaliações, esse trabalho apresenta uma análise que indica quais melhores técnicas para obtenção de uma codificação mais eficiente em termos computacionais.

O restante deste trabalho está organizado da seguinte forma:

- Capítulo 2: apresentação dos principais conceitos da compressão de vídeo, dividido em três tópicos, espaço de cores e subamostragem de cores, redundâncias de dados na representação de vídeo e modelo de codificador e decodificador de vídeo. Uma breve introdução ao padrão HEVC apontando as principais diferenças do padrão comparado ao H.264. Por fim, os principais conceitos e métodos da codificação de entropia, assim como os dois principais codificadores, o codificador CAVLC e o codificador CABAC.

- Capítulo 3: apresentação dos trabalhos correlatos os quais fazem uma comparação entre os padrões H.264 e HEVC.

- Capítulo 4: descreve a metodologia utilizada para a realização do presente trabalho.

- Capítulo 5: resultados obtidos em relação ao trabalho desenvolvido.

- Capítulo 6: considerações finais seguido das referências.

2 Revisão Bibliográfica

2.1 Conceitos Básicos da Compressão de Vídeo

Nesta seção é apresentado os principais conceitos básicos e definições da compressão de vídeo necessários para uma melhor compreensão das seções seguintes. Inicialmente será apresentado os espaços e subamostragem de cores na subseção 2.1.1. Na subseção 2.1.2 será introduzido o conceito de redundância de dados que são explorados nos compressores de vídeo, por fim, na subseção 2.1.3 será apresentado um modelo simplificado de um compressor de vídeo.

2.1.1 Espaço de Cores e Subamostragem de Cores

O vídeo digital é formado por uma sequência de quadros (*frames*) e os quadros são formados por um conjunto de *pixels*. A representação digital de um vídeo tem como referência a interpretação das cores dada pelo sistema visual humano. O olho humano possui células fotossensíveis na retina denominadas bastonetes e cones. Esses dois elementos sensíveis a luz e a cor tem como função transformar os estímulos luminosos em sinais elétricos e transmiti-los até o cérebro por meio do nervo óptico onde serão interpretados pelo cérebro como imagem (POLLACK, 2006). Cada olho possui de 6 a 7 milhões de cones. Os cones são mais sensíveis a luz do que os bastonetes e são responsáveis pela percepção das cores, do espaço e da acuidade visual. Os bastonetes por sua vez estão em um número entre 75 a 150 milhões para cada olho, localizados na superfície da retina. Diferente dos cones, os bastonetes são sensíveis a baixos níveis de iluminação não distinguindo cores e sim tons de cinza (do branco ao preto) (IIDA; WIERZZBICKI, 2005). Todas as cores interpretadas pelo sistema visual humano são resultantes da combinação entre as três cores primárias, vermelho (R - *red*), verde (G - *green*) e azul (B – *blue*), que são captadas pelos cones. Com isso, o olho humano é capaz de reconhecer milhares de tons e intensidades de cores. Porém, tratando-se de tons de cinza, que indicam a intensidade luminosa da imagem (luminância), o número é bem menor não passando de duas dúzias (GONZALEZ; WOODS, 2009).

Um vídeo digital é a representação de uma cena real. Para representar uma cor digitalmente existem várias formas. Para uma imagem monocromática (vários tons de uma única cor) necessita-se apenas de um número para indicar o brilho ou a luminância por posição de *pixel*, já nas imagens coloridas são necessários três números para ser possível representar uma cor com precisão. O método escolhido para representar o brilho a luminância e a cor é descrito como espaço de cores. Existem vários modelos que representam as cores na qual os mais utilizados são: RGB (*red, green, blue*), CMY (*cyan, magenta, yellow*), CMYK (variante do modelo CMY, onde K representa *black*), YCbCr (usado na compressão de vídeo), entre outros (FILHO; NETO, 1999). No modelo de representação de cores RGB os três valores que representam uma amostra de imagem colorida

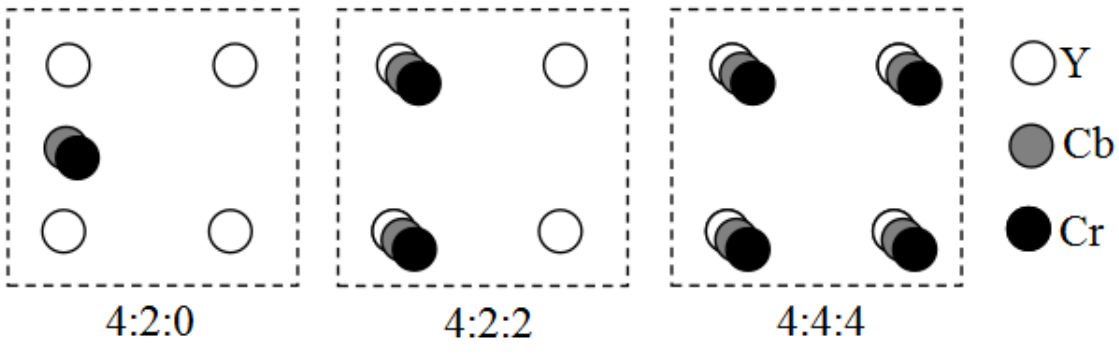
indicam as proporções das três cores primárias vermelho, verde e azul. A combinação dessas cores primárias pode resultar em todas as outras cores. O espaço de cores YCbCr possui um componente de luminância (Y), que representa a intensidade da imagem, um de crominância azul (Cb) e outro de crominância vermelha (Cr) (MIANO, 1999).

O sistema visual humano é mais sensível a luminância do que a cor. Comparando os espaços de cores RGB e YCbCr, o modelo YCbCr é o mais recomendado para aplicações envolvendo vídeo, pois seu principal componente está relacionado a luminância e separado dos componentes de crominância. Já o modelo RGB, os três componentes estão relacionados com a informação de luminância e não separados (RICHARDSON, 2011).

Os compressores de vídeo utilizam essa vantagem para diminuir as informações de crominância em relação as informações de luminância não havendo uma diferença notável, por exemplo, entre uma imagem RGB e uma YCbCr com as informações reduzidas. Este método é chamado de sub-amostragem e é usado no espaço de cores YCbCr nos compressores de vídeo (RICHARDSON, 2011).

Os formatos de amostragem mais comuns no espaço de cores YCbCr são: 4:4:4, 4:2:2 e 4:2:0, como podemos ver na Figura 1. No formato 4:4:4 para cada quatro informações de luminância (Y) temos quatro para crominância azul (Cb) e quatro para crominância vermelha (Cr), representado na Figura 1, neste padrão a sub-amostragem não foi aplicada. No formato 4:2:2 para cada quatro informações de Y na horizontal temos duas informações de Cb e duas de Cr. Por fim, no formato de amostragem 4:2:0 para cada quatro informações de Y temos uma informação de Cb e uma de Cr. Sendo assim, as amostras de crominância possuem a metade de informações na horizontal e a metade na vertical em relação as informações de luminância. Com isso, conclui-se que a sub-amostragem de cores pode reduzir até 50% dos dados considerando que o formato 4:2:0 contém 1/4 de crominância comparado ao formato 4:4:4, contendo assim, a metade de informações quando comparado ao formato 4:4:4 (RICHARDSON, 2011).

Figura 1 – Padrões de amostragem 4:2:0, 4:2:2 e 4:4:4.



Fonte: modificado de (ROSA, 2010)

2.1.2 Redundância de Dados na Representação de Vídeo

A redundância de dados é comum entre as imagens e os vídeos, pois muitos dados representam a mesma informação não sendo relevantes na representação da imagem ou do vídeo. Com isso, os algoritmos de compressão de vídeo procuram eliminar estas redundâncias para diminuir o volume de dados repetidos. Os três tipos de redundância explorados pela compressão de vídeo são: redundância espacial, redundância temporal e redundância de entrópica. Nesta seção serão apresentados os três tipos de redundância.

- **Redundância Espacial:** A redundância espacial está relacionada as informações redundantes dentro de um mesmo quadro. Um quadro (*frame*) é formado por um conjunto de *pixels*, e a redundância espacial está na correlação desses *pixels*, na qual eles tendem a terem valores similares. A diferença de informações a mais ou a menos entre os *pixels* vizinhos redundantes são muito pequenas, uma análise visual não iria distinguir a diferença. Com isso, o valor de um *pixel* pode ser calculado em relação aos *pixels* vizinhos, não precisando representar individualmente cada *pixel* contido em um mesmo *frame* (SHI; SUN, 1999). Um exemplo simples de redundância espacial é um avião sobrevoando o céu sem nuvens na qual a informação mais importante é o avião, já o fundo, onde a maior parte é azul, as informações são praticamente uniformes (YAMADA et al., 2010).

- **Redundância Temporal:** A redundância temporal está relacionada as informações similares ou iguais entre quadros temporalmente vizinhos. Um vídeo é uma sequência de quadros que possuem uma grande correlação entre eles tendo vários *pixels* com a mesma intensidade de cor ou com uma grande similaridade, tendo assim, pouca mudança de informações entre os quadros anteriores ou posteriores (exceto em cortes de cena) (SHI; SUN, 1999). Um exemplo prático de redundância temporal seria a imagem de um jogador de golfe em uma partida. Após a sacada, a diferença entre os quadros sucessivos, seria apenas em relação a posição da bola, o restante da imagem, é considerado informações redundantes (YAMADA et al., 2010).

- **Redundância Entrópica:** A redundância de entropia difere-se das outras redundâncias (espacial e temporal) pelo fato de não estar relacionada diretamente com o conteúdo da imagem, mas sim com a representação das informações (SHI; SUN, 1999). Eliminando a redundância de entropia não se tem perda de dados, com isso, os algoritmos têm como objetivo transmitir o maior número de dados possíveis por símbolo codificado tendo como resultado a representação de um maior número de informações com um menor número de bits (AGOSTINI, 2007). Existem vários algoritmos que realizam a codificação de entropia na qual são apresentados no decorrer deste trabalho.

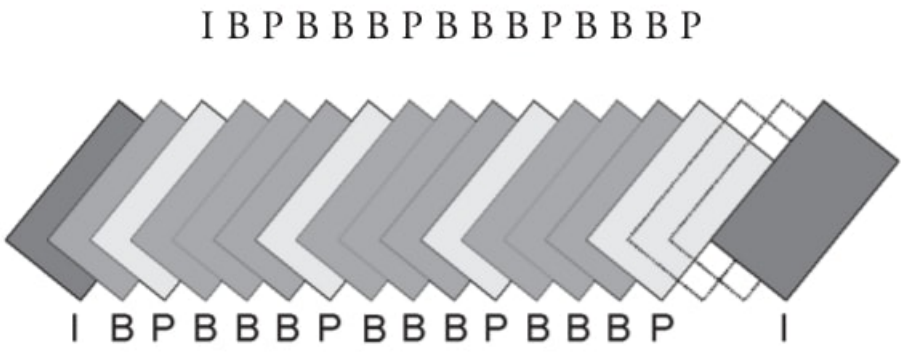
2.1.3 Modelo de Codificador e Decodificador de Vídeo

A compressão de vídeo é composta por dois sistemas, um compressor (codificador) e um descompressor (decodificador). O compressor tem como objetivo reduzir os dados que representam um vídeo original, representando-o em um número menor de bits antes do armazenamento ou da transmissão. O decodificador executa o processo inverso, convertendo os dados reduzidos para a representação do vídeo original (RICHARDSON, 2011). O resultado da compressão do vídeo advém da eliminação das redundâncias que foram apresentadas na subseção 2.1.2.

Um modelo de compressor de vídeo é apresentado na Figura 3. São utilizados neste modelo dois quadros simultaneamente, um é o quadro que está sendo codificado (comprimido) e o outro o quadro de referência. Existem vários tipos de quadros na compressão de vídeo. No modelo apresentado nesta seção serão utilizados apenas o quadro I e o quadro P, mas para um melhor entendimento, na Figura 2 é apresentado os três principais tipos de quadros. A representação das letras estão descritas abaixo:

- *I - Intra Frame:* os quadros *intra frames* são quadros completos utilizados no começo da sequência (YAMADA et al., 2010).
- *B - Bidirectional Frames:* são quadros que possuem diferença entre o quadro atual e os quadros antecessores ou sucessores (YAMADA et al., 2010).
- *P - Predicted Frames:* são quadros que adquirem informações do quadro antecessor, possuem uma alta quantidade de informações comprimidas e também podem ser usados como referência para o quadro sucessor (YAMADA et al., 2010).

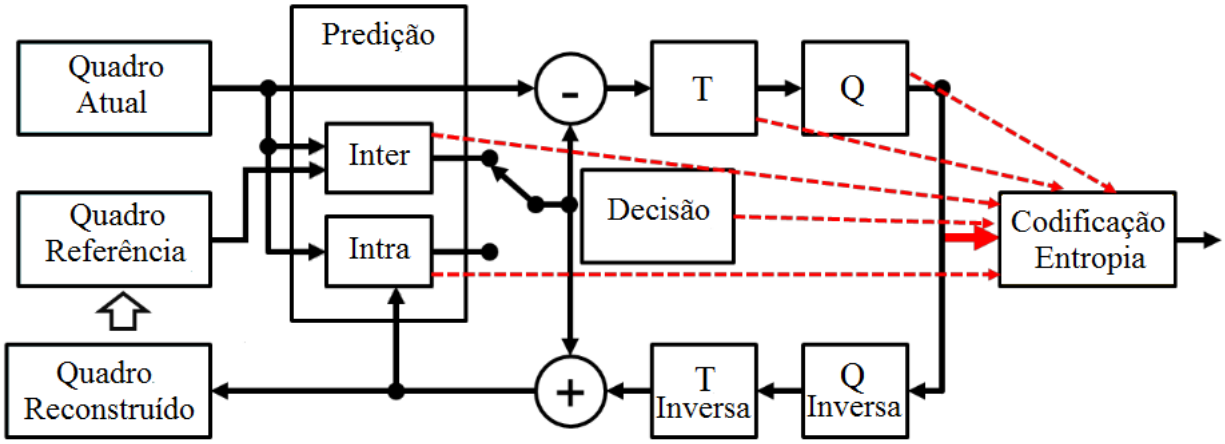
Figura 2 – Transmissão de quadros em um modelo de compressão de vídeo.



Fonte: (YAMADA et al., 2010)

Na compressão de vídeo os quadros são divididos e subdivididos em blocos. Os blocos do quadro atual na Figura 3, que é o quadro que está sendo codificado, pode seguir

Figura 3 – Modelo de compressor de vídeo.



Fonte: modificado de (THIELE, 2012)

dois caminhos, ou ele é codificado pela codificação intra-quadro ou pela codificação inter-quadros. O modelo de compressor de vídeo apresentado na Figura 3 é constituído pelos módulos de predição intra-quadro, predição inter-quadros, transformada, quantização e codificação de entropia. Uma explicação básica dos módulos do compressor de vídeo estão descritas a seguir:

- **Codificação Intra-Quadro:** a codificação intra-quadro tem como objetivo reduzir a redundância espacial eliminando a redundância apenas dos blocos do quadro atual que está sendo processado. Após o quadro atual passar pela codificação intra-quadro o resultado são blocos do tipo I que poderão ser usados como referência na codificação dos quadros do tipo I e P.

- **Codificação Inter-Quadros:** na codificação inter-quadros o objetivo é reduzir a redundância temporal entre os blocos dos quadros vizinhos, como por exemplo, entre os blocos do quadro atual e os blocos do quadro de referência, representados na Figura 3. Existem duas etapas nessa codificação, a estimação de movimento e a compensação de movimento. Diferente da codificação intra-quadro, na codificação inter-quadros os blocos resultantes são do tipo P e são utilizados por quadros do tipo P. Dos blocos resultantes, aquele que tiver uma maior semelhança com o bloco do quadro de referência é o selecionado. Por fim, após todo o processo, a codificação inter-quadros gera um vetor de movimento para identificar o bloco selecionado no quadro de referência.

  Após o bloco do quadro atual passar por uma dessas codificações apresentadas anteriormente, inter-quadros ou intra-quadro, obtêm-se o resíduo, que é o resultado da subtração entre as informações do quadro original e os valores gerados pelas codificações.

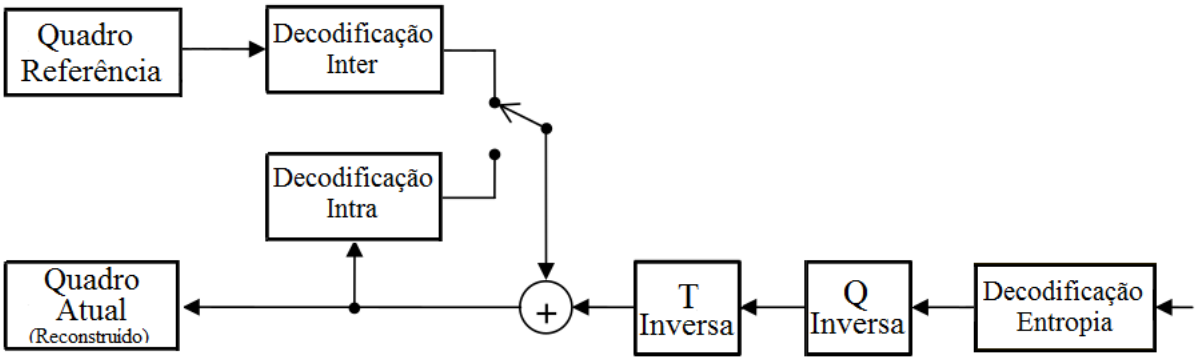
- **Transformada e Quantização:** dando sequência no processo de compressão de vídeo, após ser gerado o resíduo, o mesmo passa pela transformada (bloco T na Figura 3) que transforma as informações no domínio espacial em informações no domínio das frequências. Após as informações estarem no domínio das frequências a quantização pode ser aplicada com um melhor aproveitamento, reduzindo assim, a redundância espacial, eliminando os dados menos relevantes ao olho humano.

- **Codificação de Entropia:** por fim, é aplicada a codificação de entropia que tem como objetivo reduzir a redundância entrópica. Existem vários algoritmos voltados para codificação de entropia e alguns serão apresentados no decorrer deste trabalho.

- **Transformada Inversa e Quantização Inversa:** outro processo importante no codificador de vídeo é a quantização inversa e a transformada inversa. Os dois tem como objetivo gerar o quadro atual reconstruído que seria uma parte do decodificador inserida no codificador. Este quadro atual reconstruído serve como referência para as codificações intra-quadro e inter-quadros. Com isso, após o quadro atual ser codificado, ele é descartado e o quadro reconstruído toma seu lugar como referência. Assim, o codificador terá a mesma referência que o decodificador, evitando erros na decodificação e aumentando a precisão na reconstrução da imagem no decodificador. Este processo não é viável para a codificação de entropia, pois a codificação entrópica não gera perdas.

Na Figura 4 temos o modelo de decodificação de vídeo. Processo semelhante ao caminho de reconstrução na codificação de vídeo (Figura 3).

Figura 4 – Modelo de descompressor de vídeo.



Fonte: modificado de (AGOSTINI, 2007)

A decodificação de entropia reconstrói os dados codificados pelo codificador. Como a codificação de entropia não insere perdas, o resultado após a decodificação é igual aos dados resultantes após o processo de quantização no codificador.

Após a decodificação de entropia, os dados reconstruídos passam pela quantização inversa e pela transformada inversa gerando assim o resíduo. Esse processo é semelhante a reconstrução apresentada no codificador.

Na decodificação inter-quadros são utilizados o quadro de referência e as informações de controle (que não são mostradas na Figura 4). As informações de controle localizam os blocos no quadro de referência, que posteriormente, são somados aos resíduos, resultando os blocos reconstruídos. Por fim, esses blocos reconstruídos são armazenados para serem utilizados como referência para os próximos quadros e também para a versão final do vídeo decodificado (AGOSTINI, 2007).

Na decodificação intra-quadro são utilizados os blocos do quadro atual reconstruído e também as informações de controle (assim como na codificação inter-quadros, não é mostrado as informações de controle na imagem), para assim reconstruir os blocos que foram codificados pelo codificador intra-quadro. Os blocos resultantes da decodificação intra-quadro são somados aos resíduos e armazenados para serem usados como referência e também para a visualização do vídeo, assim como na decodificação inter-quadros (AGOSTINI, 2007).

2.2 Padrão HEVC

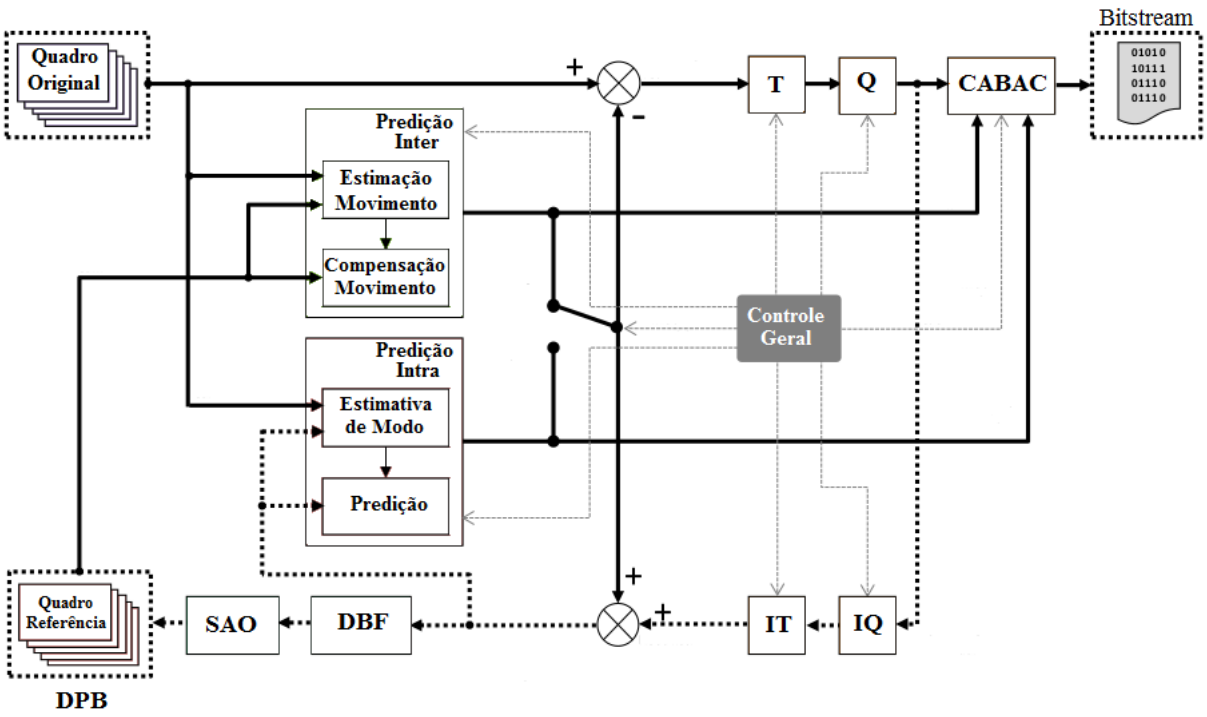
Como a resolução e a exigência por qualidade dos vídeos digitais vem aumentando, acaba sendo necessário também a evolução dos compressores de vídeo. O padrão estado-da-arte de codificadores de vídeo é o HEVC, sucessor do padrão H.264. Segundo (SULLIVAN et al., 2012) o compressor HEVC pode chegar ao dobro de eficiência quando comparado ao H.264. Nesta seção será apresentada uma breve introdução do padrão HEVC e já apontando as principais diferenças do padrão comparado ao H.264.

2.2.1 Modelo de Compressor de Vídeo HEVC

A Figura 5 mostra o diagrama em blocos do codificador de vídeo HEVC. Os blocos que compõem o codificador são semelhantes aos dos padrões anteriores. O codificador é composto pelos seguintes módulos: Predição intra-quadro, inter-quadros, transformada, quantização, transformada inversa, quantização inversa, codificação de entropia e filtro de blocagem. No decorrer desta seção será explicado cada um dos módulos apresentados anteriormente.

O padrão HEVC possui o esquema de codificação híbrida baseada em bloco, assim como os padrões de codificação de vídeo anteriores. Uma das principais inovações do padrão HEVC, que contribui para um melhor desempenho, são os mecanismos de blocos de tamanhos maiores e também uma melhor flexibilidade nas partições e subpartições dos mesmos (POURAZAD et al., 2012). Diferente do padrão H.264 o padrão HEVC pode ter um bloco de entrada de tamanho 64x64. O bloco de entrada pode ser dividido em CTUs

Figura 5 – Modelo de compressor de vídeo HEVC.



Fonte: modificado de (CORREA, 2015)

(*Coding Tree Units*) que pode ter tamanho 64x64 *pixels* ou menor. AS CTUs também podem ser divididas em CUs (*Coding Units*), na qual o seu tamanho mínimo é de 8x8 *pixels*. As CUs também podem ser subdivididas em pedaços ainda menores, chamados de unidades de predição PUs (*Prediction Units*), na qual o seu tamanho mínimo é de 4x4 *pixels* e o máximo igual ao tamanho da CU (SILVEIRA et al., 2016). Nas seções seguintes são apresentados os módulos da codificação de vídeo do padrão HEVC.

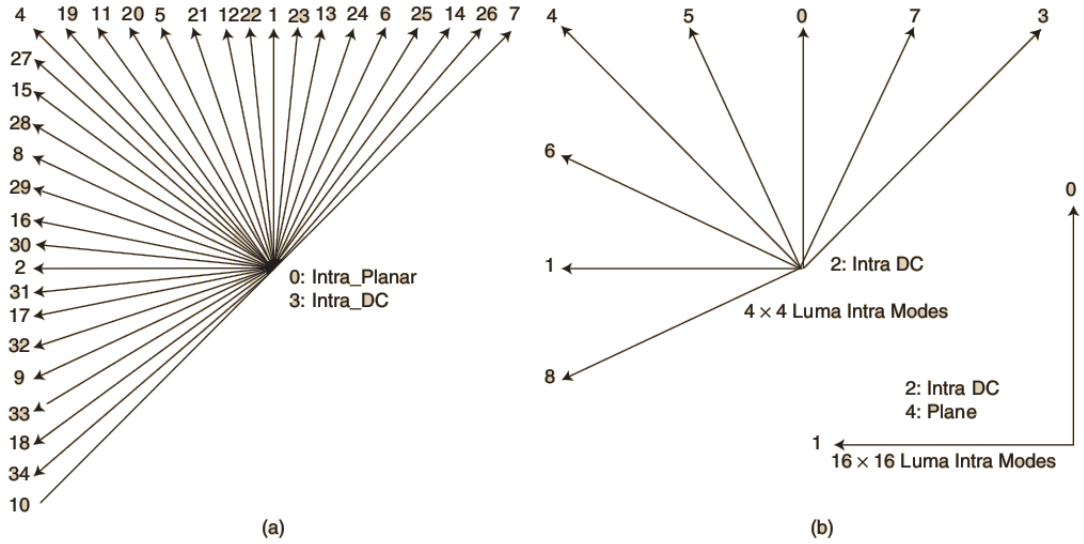
2.2.1.1 Predição Intra-Quadro

O módulo de predição intra-quadro tem como objetivo reduzir a redundância espacial. Neste processo é gerado a predição de blocos candidatos que são baseados nas informações dos blocos vizinhos da imagem atual e também dos blocos que já foram processados e reconstruídos (SILVEIRA et al., 2016).

O módulo de predição intra no padrão HEVC possui mais módulos de predição comparado ao padrão H.264. O padrão HEVC utiliza 35 modos de predição, 33 modos de predição angular (direcionais) e mais as predições DC e planar. Na Figura 6 é possível visualizar os 35 modos de predição intra do padrão HEVC e também mostra os módulos de predição do padrão H.264, sendo possível verificar a diferença entre os dois padrões. No padrão HEVC o modo que apresenta mais eficiência, entre os 35, é escolhido. Para a seleção do modo é considerado os valores da taxa de distorção (RD) (SILVA, 2015).

Algumas métricas são utilizadas na etapa de decisão, como a SATD e SAD.

Figura 6 – Modos de predição intra dos padrões (a) HEVC e (b) H.264.

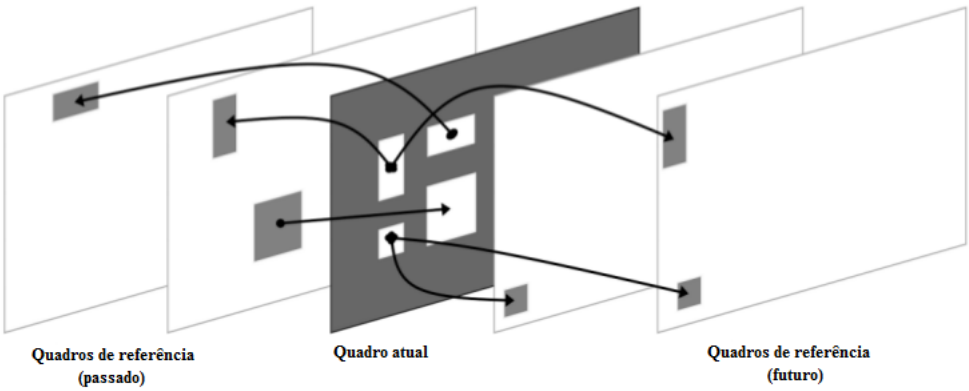


Fonte: modificado de (POURAZAD et al., 2012)

2.2.1.2 Predição Inter-Quadros

O módulo de predição inter do padrão HEVC, assim como no padrão H.264, é composto pelos blocos de estimação de movimento (ME) e compensação de movimento (MC). A predição inter tem como objetivo reduzir a redundância temporal daqueles dados redundantes entre os quadros vizinhos. A estimação de movimento procura os blocos que possuem a maior semelhança comparando os blocos do quadro atual e o quadro de referência. Um exemplo do processo de predição inter pode ser visualizado na Figura 7.

Figura 7 – Previsão do quadro atual em relação aos quadros de referência.



Fonte: modificado de (CORREA, 2015)

Após ser realizada a estimação de movimento, tem-se como resultado o vetor de movimento (MV – *Motion Vector*). O vetor de movimento indica o deslocamento entre a posição do bloco atual e o bloco escolhido na imagem de referência. Assim como nos padrões anteriores, o vetor de movimento, após ser gerado, é codificado pelo codificador de entropia e enviado pelo *bitstream* junto com o resíduo. A compensação de movimento realiza o processo inverso da estimação de movimento, tendo como resultado o bloco reconstruído, utilizando assim o vetor de movimento. Por isso o bloco de compensação de movimento também é utilizado no decodificador de vídeo (SILVEIRA et al., 2016).

2.2.1.3 Transformada e Quantização

O módulo da transformada (bloco T, na Figura 5) tem como objetivo transformar as informações do resíduo em domínio da frequência. Após as informações do resíduo passarem pela transformada, elas são processadas pelo módulo da quantização (bloco Q, na Figura 5). Semelhante ao padrão H.264, o HEVC usa transformações inteiras baseadas em Transformadas Cosseno Discretas (DCT), porém, com tamanhos de blocos diferentes que variam de 4x4 a 32x32. No HEVC os blocos também podem ser retangulares (não quadrados), na qual as transformações de linhas e colunas possuem tamanhos diferentes (POURAZAD et al., 2012). Apenas as matrizes de transformações 32x32 do HEVC são definidas, o restante das matrizes são derivadas da transformação da matriz 32x32 usando parte de suas entradas. O HEVC também possui uma transformada alternativa com base na transformada de seno discreta (DST) para os blocos dos resíduos de luminância 4x4 resultantes da predição intra (CORREA, 2015). Os coeficientes resultantes da transformada passam pela quantização com o objetivo de reduzir as informações que não são relevantes para o sistema visual humano. No processo de quantização, com a redução dos dados, as perdas são irreversíveis, e essas perdas são controladas pelo parâmetro de quantização (QP). No caso de QP constante e taxa de bits variável, o QP é definido na configuração do codificador. Porém, no casdo de QP variável e taxa de bits constante, o QP é definido por um algoritmo de controle de taxa de codificação (SILVA, 2015).

2.2.1.4 Codificação de Entropia

A codificação de entropia é a etapa que ocorre após a quantização e tem como objetivo codificar todos os dados sintáticos e coeficientes transformados quantizados. Diferente dos outros blocos a codificação entrópica não possui perda de informações.

No padrão H.264 a codificação de entropia possui dois tipos de codificações: a CAVLC (Codificação de comprimento variável adaptativa ao contexto) e o CABAC (Codificação Aritmética Binária Adaptativa ao Contexto). O CABAC possui uma melhor eficiência de codificação que o CAVLC, porém, o CABAC aumenta a complexidade de codificação.

No padrão HEVC é utilizado apenas o CABAC para codificar todos os dados do codificador de vídeo. O CABAC do HEVC é bem semelhante com o CABAC do H.264, porém, sofreu algumas modificações. A quantidade de contexto do CABAC utilizada no HEVC é menor que a quantidade utilizada no H.264, no entanto, o desempenho de compressão é melhor e mais rápido. No HEVC os modelos de contexto são obtidos através das informações das CUs vizinhas, assim como os particionamentos para as decisões de profundidade das árvores de CUs e TUs, servem para gerar índices de modelo de contexto (SILVA, 2015).

Assim como no padrão H.264, o HEVC possui alguns processos de binarização: Unário (U), Unário Truncado (TU), Exponencial Golomb (EXPG) e comprimento fixo (FL) (SZE; BUDAGAVI, 2012). Os exemplos dos códigos binarizados podem ser vistos na Tabela 1.

A codificação aritmética pode ser executada de duas formas, usando uma probabilidade estimada (contexto codificado) ou utilizando uma probabilidade igual a 0.5 (codificada como *bypass*). Para os *bins* codificados pelo processo *bypass*, a divisão do intervalo em subintervalos pode ser feita por um deslocamento de bits, já para os *bins* codificados no contexto, é utilizada uma tabela de consulta (SZE; BUDAGAVI, 2012).

Tabela 1 – Exemplos de padrões de binarização utilizados no HEVC.

N	Unário (U)	Unário Truncado (TU)	Exponencial Golomb (EXPG)	Comprimento Fixo (FL)
0	0	0	1	000
1	10	10	010	001
2	110	110	011	010
3	1110	1110	00100	011
4	11110	11110	00101	100
5	111110	111110	00110	101
6	1111110	1111110	00111	110
7	11111110	1111111	0001000	111

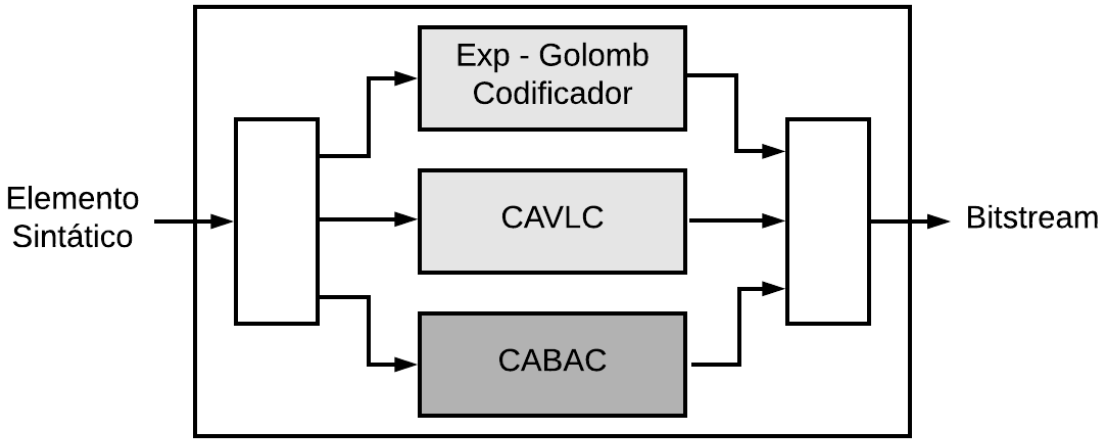
Fonte: modificado de (SZE; BUDAGAVI, 2012)

2.3 Processo de Codificação de Entropia

A codificação de entropia tem como objetivo reduzir a redundância entrópica dos símbolos que se encontram nos blocos do resíduo após a transformada e a quantização. A codificação de entropia também é responsável pela codificação de elementos sintáticos de controle e da configuração do decodificador. Essas informações resultam da predição, transformada e quantização. Também é codificado pelo codificador de entropia o que cada um desses módulos escolheu como opção para a codificação da imagem. Existem

vários métodos para auxiliar na codificação de entropia, dependendo do padrão utilizado. Nesta seção será utilizado como exemplo o modelo de codificador de Entropia conforme o padrão H.264, como mostra a Figura 8.

Figura 8 – Modelo de codificador de entropia conforme o padrão H.264.



Fonte: modificado de (DEPRA, 2009)

A Figura 8 contém os três blocos que compõem o codificador de entropia, na qual cada um deles representa:

- **Código de Comprimento Fixo:** no código de comprimento fixo é feita a conversão do símbolo para um código binário que tem o comprimento especificado.
- **Código de Comprimento Variável Exponencial Golomb (EXPG):** cada símbolo é representado por uma palavra de código EXPG com o número de bits variável. Normalmente nesse método, as palavras códigos EXPG mais curtas representam os símbolos que aparecem com mais frequência.
- **Codificação de Comprimento Variável Adaptativa ao Contexto (CAVLC):** o método de codificação CAVLC codifica coeficientes de transformada em que diferentes conjuntos de códigos de comprimento variável são escolhidos dependendo da estatística dos coeficientes recentemente codificados usando a adaptação de contexto.
- **Codificação Aritmética Binária Adaptativa ao Contexto (CABAC):** no método de codificação aritmética CABAC os modelos de probabilidade são atualizados com base nas estatísticas de codificação anteriores.

Todos os métodos apresentados anteriormente têm como objetivo codificar os símbolos, reduzindo a redundância entrópica, convertendo esses símbolos em um padrão binário que é transmitido ou armazenado como parte do fluxo de bits, o *bitstream*.

No caso dos elementos sintáticos que estão nos níveis superiores (GOP, quadros, etc.) são codificados usando códigos binários fixo ou de comprimento variável. Nos níveis de *slices* ou abaixo (macroblocos, blocos, etc.), a codificação dos elementos sintáticos é feita pela codificação aritmética binária adaptativa ao contexto (RICHARDSON, 2011) e também por códigos de comprimento variável (SALOMON, 2004). Nas informações dos resíduos, que são os coeficientes das transformadas quantizados, é utilizado a codificação de comprimento variável adaptativa ao contexto (RICHARDSON, 2011), e para o resto dos elementos sintáticos de outras unidades é utilizado códigos Exp-Golomb (SALOMON, 2004).

Antes de prosseguir com a explicação referente a codificação de entropia, será apresentado uma breve explicação em relação a transformada hadamard, para um melhor entendimento do restante da seção e do trabalho apresentado.

A transformada direta (T) no compressor de vídeo recebe blocos 4x4 resultantes da predição. Para realizar a transformada é aplicada a FDCT 2-D para todas as amostras de luminância e crominância. Porém, no caso das informações de luminância e crominância da predição intra-quadro 16x16, é aplicado também a transformada Hadamard 2-D. Sobre os coeficientes resultantes da FDCT 2-D.

Nas informações de luminância é aplicada a transformada Hadamard 4x4 direta, e para as informações de crominância é aplicada a transformada Hadamard 2x2 direta.

A transformada Hadamard 4x4 é apresentada na Equação 2.1, na qual o resultado é uma matriz 4x4 WD.

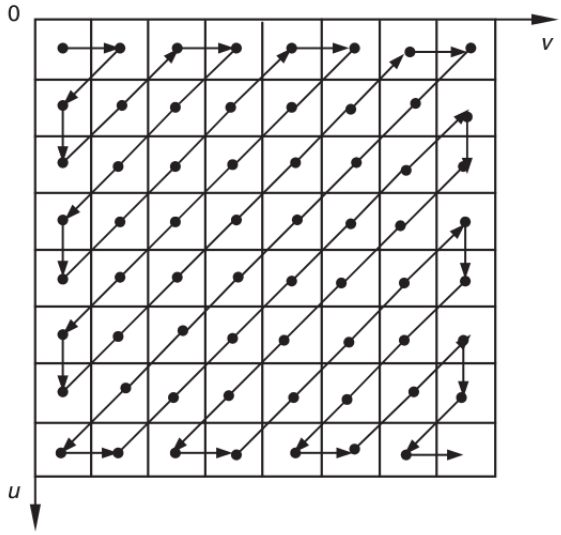
$$Y_D = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} \\ W_D \\ \\ \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \right) /2 \tag{2.1}$$

A transformada Hadamard 2x2 é apresentada na Equação 2.2, essa transformada é aplicada em Cb e Cr resultando em uma matriz 2x2.

$$Y_{QD} = \left(\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} W_D \\ \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \right) \tag{2.2}$$

Antes da codificação de entropia, as informações residuais resultantes da quantização, como os blocos 4x4 com coeficientes das transformadas, passam por uma reordenação em forma de ziguezague, como mostra a Figura 9. As amostras resultantes da transformada hadamard 4x4 e 2x2 também passam pela reordenação em ziguezague, porém, começam pela segunda posição em vez da primeira e possuem um elemento a menos na matriz, 15 em vez de 16, por consequência da própria transformada hadamard.

Figura 9 – Ordem ziguezague de leitura dos blocos 4x4 para a codificação de entropia.



Fonte: (SHI; SUN, 1999)

A seguir serão apresentados os três principais métodos utilizados na codificação de entropia (EXPG, CAVLC e CABAC), na qual os mesmos já tiveram uma breve apresentação anteriormente.

2.3.1 Código de Comprimento Variável Exponencial Golomb (EXPG)

A codificação exponencial Golomb são códigos binários de tamanho variável com um padrão regular a ser seguido. A codificação EXPG considera a probabilidade dos dados. As informações simbólicas mais frequentes são representadas por um código binário menor, já aquelas informações simbólicas menos frequentes são representadas por um código binário maior. Na Equação 2.3 pode-se observar a estrutura dos dados já codificados pela codificação EXPG:

$$[\text{Prefixo Zero}]\,[1]\,[INFO] \tag{2.3}$$

Essa estrutura consiste em um prefixo de M zeros, um campo com a informação 1, e outro com a informação M-bits, INFO. O prefixo zero (M zero) corresponde ao número de zeros que antecedem o campo com a informação 1. Onde M é representado na Equação 2.4

$$M = log_2\,(cod\_num + 1) \tag{2.4}$$

Na Equação 2.4, *cod_num* é o elemento a ser codificado, e na Equação 2.5 o campo INFO possui M bits informando a informação codificada, na qual o mesmo é dado por:

$$INFO = cod\_num + 1 - 2^M \tag{2.5}$$

Na Tabela 2, podemos observar as primeiras informações codificadas pela codificação EXPG, pode-se notar que o primeiro valor codificado possui apenas o campo 1, já o restante das informações contém todos os campos da estrutura do código EXPG.

Tabela 2 – Primeiros códigos EXPG.

codnum	Código
0	1
1	010
2	011
3	00100
4	00101
5	00110
...	...

Fonte: modificado de (AGOSTINI, 2007)

2.3.2 Codificação de Comprimento Variável Adaptativa ao Contexto (CAVLC)

O método CAVLC tem como objetivo codificar as informações residuais resultantes da quantização. Antes de ser realizada a codificação as informações do resíduo passam pela reordenação em ziguezague, método no qual foi explicado anteriormente, como mostra a Figura 9.

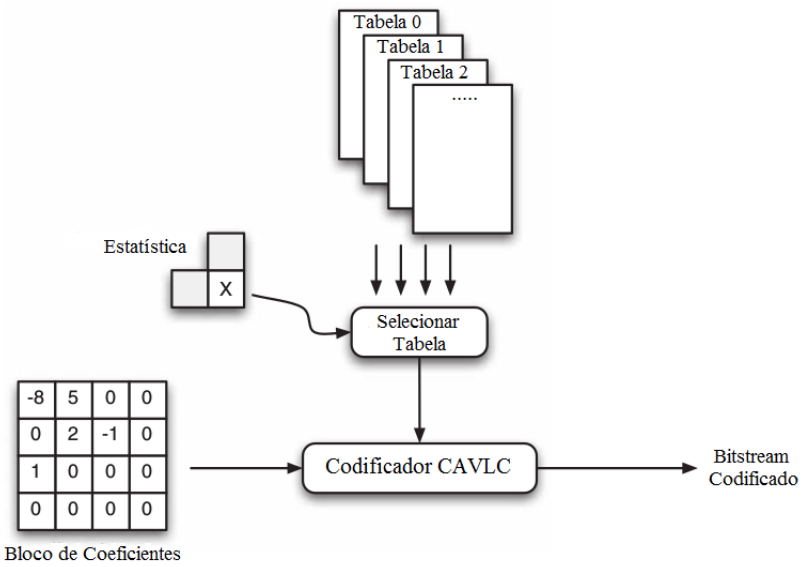
Após serem reordenadas, as informações passam pelo processo de codificação, na qual são realizados cálculos e consulta a tabelas com códigos de comprimento variável. Os parâmetros que são mais frequentes serão convertidos em códigos menores.

Na Figura 10 é mostrado o processo de codificação CAVLC simplificado. As informações são convertidas em códigos de comprimento variável. Tabelas VLCs são selecionadas com base nos números de coeficientes não nulos em blocos vizinhos e também nos coeficientes recentemente codificados. O parâmetro nC, que considera os valores não zeros dos blocos vizinhos (bloco superior e bloco da esquerda), é obtido da seguinte forma: quando os blocos vizinhos estão presentes, nC é a média entre eles, quando apenas um bloco está presente, esquerdo ou superior, nC recebe o valor do bloco existente, já se os dois blocos não existem, o valor de nC é igual a zero (CHIMURKAR, 2015).

Após o codificador CAVLC receber o vetor de coeficientes e o valor de nC, ele está pronto para extrair os dados. Cada vetor é codificado pela CAVLC nas seções de dados seguintes:

- *Coefficient Token:* total de amostras não zeros dentro do bloco e de coeficientes de fuga no vetor atual (CHIMURKAR, 2015).

Figura 10 – Visão geral do codificador CAVLC.



Fonte: modificado de (RICHARDSON, 2011)

- *Trailing Ones:* é determinada utilizando algoritmos e representa o sinal de cada trailing ones (CHIMURKAR, 2015). Como por exemplo, se o traling ones for +1, ele é codificado como 1, se o mesmo for –1, será codificado como 1, se não existir trailing ones, este elemento sintático não é codificado (THIELE, 2012).

- *Level:* este elemento sintático está relacionado aos elementos não zeros e que também não são trailing ones (CoeffLevel) no vetor atual, excluindo os que estão à direita (CHIMURKAR, 2015). Não existindo coefflevel no bloco, a codificação não é aplicada. Este elemento possui um prefixo <nzeros 1> e um sufixo. No cálculo para determinar o prefixo e o sufixo, quando coefflevel tem um valor positivo o resultado é 0, porém quando o valor é negativo, o resultado é 1. Na sequência é feito um cálculo para saber o tamanho do próximo sufixo baseado no valor de coefflevel (THIELE, 2012).

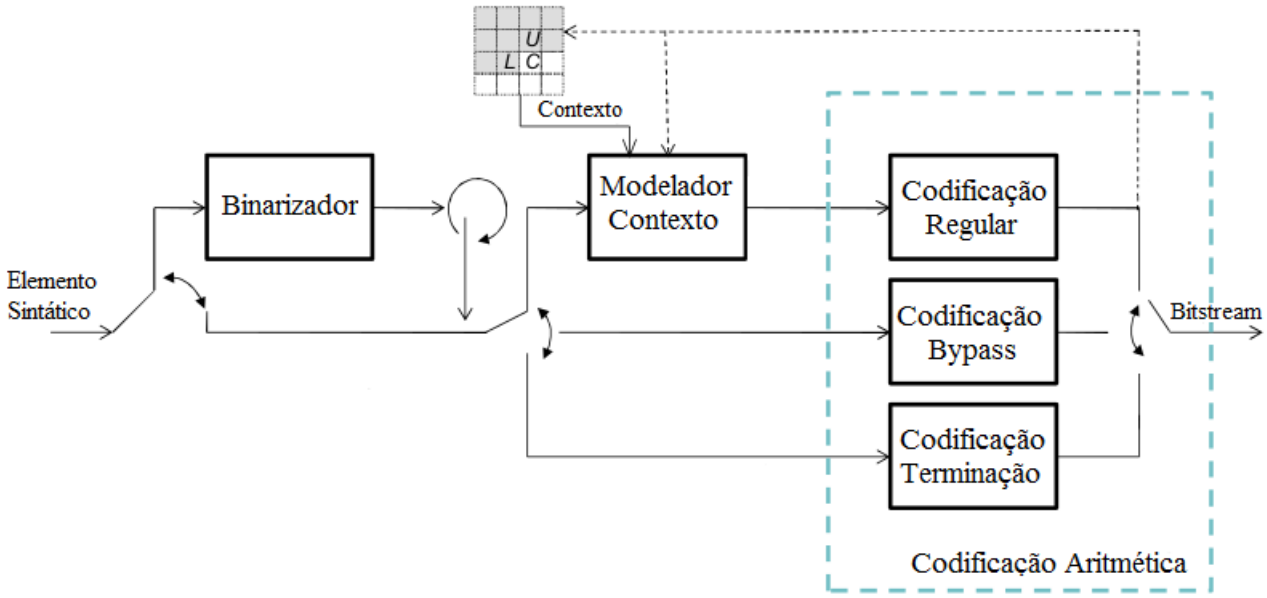
- *Totalzeros:* total de zeros antes do primeiro não zero.

- *Run Before:* este elemento está relacionado com a distribuição de zeros entre o primeiro e o último elemento não zero do bloco (CHIMURKAR, 2015).

2.3.3 Codificação Aritmética Binária Adaptativa ao Contexto (CABAC)

Diferente dos demais métodos de codificação apresentados nessa seção, o CABAC tem como objetivo codificar os elementos sintáticos a nível de macrobloco, e disponível apenas para os perfis main e hight (no padrão H.264). O diagrama de blocos da codificação CABAC está representado na Figura 11.

Figura 11 – Visão geral do codificador CABAC.



Fonte: modificado de (KIM et al., 2009)

O CABAC é dividido em três principais etapas: binarização, seleção do modelo de contexto e codificação aritmética.

2.3.3.1 Binarização

O CABAC utiliza codificação aritmética binária, isso quer dizer que apenas valores binários são codificados. Com isso, todos os valores não binários são convertidos em códigos binários antes de passarem pela codificação aritmética. Esse processo de conversão é semelhante a conversão dos dados em códigos de comprimento variável, visto anteriormente. Porém, no caso do CABAC, os códigos convertidos ainda passam pela codificação aritmética antes da geração do *bitstream*. Nos códigos binários, cada posição desses dados é chamada de *bin*.

2.3.3.2 Seleção do Modelo de Contexto

Um modelo de contexto é um modelo que contém a probabilidade de ocorrência dos *bins*, podendo ser apenas para um ou para vários. É feita uma seleção de alguns modelos disponíveis na qual a escolha é feita em relação aos dados que foram codificados recentemente, portanto, estes dados são atualizados conforme o decorrer da codificação. Após a escolha do modelo, o mesmo armazena a probabilidade de cada *bin* ser 1 ou 0.

Existe também uma atualização do modelo de contexto escolhido, que tem como base o valor que está sendo codificado. Por exemplo, se o valor do *bin* é 1, será aumentada a contagem da possibilidade do valor 1.

2.3.3.3 Codificação Aritmética

A codificação aritmética é o módulo que gera o resultado final do CABAC. Resumidamente, a codificação aritmética transforma cada *bin* tendo como referência o modelo de contexto escolhido.

Após os dados passarem pelo processo de binarização e ser escolhido o modelo de contexto mais apropriado, os dados passam pela codificação aritmética. Este módulo possui três mecanismos de codificação: os mecanismos regular, *bypass* e *terminate*. Uma explicação básica dos mecanismos de codificação estão descritas abaixo:

- O modo regular de codificação utiliza os modelos de contexto para definir os limites de valores de intervalo e também para o deslocamento do cálculo aritmético.
- O modo de codificação *bypass* codifica aqueles valores que possuem a mesma probabilidade de ocorrência. Diferente do modo regular, o modo *bypass* não utiliza o modelo de contexto e nem a normalização do intervalo. Nesse modo, é feita somente a atualização dos limites superiores e inferiores, baseado nos valores dos dados codificados (DEPRA, 2009).
- Por fim, o modo de codificação *terminate* codifica os elementos que marcam o final da codificação de um *slice*. Esse modo possui probabilidade fixa não sendo lido ou atualizado o modelo de contexto (DEPRA, 2009).

3 Trabalhos Relacionados

Neste capítulo, serão apresentados os principais trabalhos relacionados, que possuem relevância com a proposta do presente trabalho. Entre os trabalhos relacionados, o primeiro apresenta uma comparação geral entre os padrões HEVC e H.264, e os outros dois uma comparação focada no bloco de codificação de entropia.

3.1 Benchmarking the Encoding Efficiency of H.265/HEVC and H.264

O trabalho de (KOUMARAS; KOURTIS; MARTAKOS, 2012) tem como objetivo avaliar a eficiência de codificação e compressão do padrão HEVC. Foi realizada uma comparação entre o HEVC e o H.264, validando o objetivo principal do padrão HEVC de dobrar a eficiência compressiva comparado ao seu antecessor, sem redução no nível de qualidade visual do vídeo codificado.

Foram utilizados três vídeos de referência para os testes no trabalho, na qual foram representados pelos nomes: Bubbles, Horse Race, Quadrado BQ. Os sinais de teste tem resolução de 416x240, e o formato YUV original não codificado. Os vídeos originais são codificados para ISO AVC Baseline Profile, e para os perfis HEVC: Perfil de Acesso Aleatório (RAP), Perfil de Baixa Complexidade de Acesso Aleatório (RALCP), e Perfil de atraso baixo (LDP). Em todo processo de codificação foram utilizados softwares de referência do H.264 e HEVC, no caso do HEVC, especificadamente, foi utilizado o software (HM)1.0.

Foram feitos três tipos de testes no trabalho de (KOUMARAS; KOURTIS; MARTAKOS, 2012), o tempo de duração do processo de codificação dos vídeos, a eficiência da compressão e a eficiência da codificação do HEVC e H.264.

Na Tabela 3 está exposto os resultados da duração do processo de compressão dos vídeos para os três perfis HEVC em relação a duração do processo de compressão do perfil H.264.

Tabela 3 – Tempo de codificação dos perfis do HEVC em comparação ao H.264.

-	RAP	RALCP	LDP
A	4.85	4.45	3.76
B	3.99	3.84	3.46
C	3.98	3.80	3.70

Fonte: modificado de (KOUMARAS; KOURTIS; MARTAKOS, 2012)

Conforme as informações da Tabela 3, pode-se deduzir que a duração do processo HEVC é mais elevada que a duração do processo H.264. Segundo o trabalho de (KOUMARAS; KOURTIS; MARTAKOS, 2012), a duração do processo de codificação do HEVC, em média, é quatro vezes maior que a duração do processo do H.264.

Na Tabela 4 está apresentado a eficiência do processo de compressão do HEVC em comparação ao H.264. É possível observar com os resultados descritos na tabela, que os dados codificados em HEVC são aproximadamente 50% mais eficientes comparados ao H.264.

Tabela 4 – Melhora na eficiência de compressão do HEVC comparado ao H.264.

-	RAP	RALCP	LDP
A	49%	44%	30%
B	66%	49%	36%
C	51%	48%	35%

Fonte: modificado de (KOUMARAS; KOURTIS; MARTAKOS, 2012)

Finalmente, na comparação da eficiência de codificação entre o HEVC e H.264 pelo uso da métrica PSNR, teve como resultado que a eficiência do HEVC é semelhante a do padrão H.264, porém, dobrando a eficiência de compressão do fluxo de bits. Detalhando melhor os resultados, o perfil LDP do HEVC apresentou resultados piores em relação ao H.264, assim como o valor PSNR médio, inferior em comparação com o H.264. Já os perfis RAP e RALCP, apresentaram resultados melhores ou similares comparados ao H.264.

O trabalho de (KOUMARAS; KOURTIS; MARTAKOS, 2012) confirmou com os testes, comparações e resultados, que o compressor de vídeo HEVC atinge o dobro da eficiência comparado ao H.264, resultado na qual era o principal objetivo do trabalho em relação ao padrão.

3.2 High Throughput CABAC Entropy Coding in HEVC

O trabalho de (SZE; BUDAGAVI, 2012) tem como objetivo mostrar as principais técnicas usadas para melhorar o CABAC do HEVC em relação a codificação de entropia do padrão H.264, assim como reduções dos requisitos de memória que de alguma forma favoreceram no custo de produção e implementação. Além disso, foram realizados testes e simulações com o novo CABAC do HEVC, comparando ao CABAC do H.264. Algumas informações importantes foram discorridas no trabalho. O CABAC é um gargalo de taxa de transferência. O rendimento do CABAC é analisado com base na quantidade de informações binárias processadas por segundo. Existem muitas técnicas que podem melhorar o rendimento do CABAC, porém, a dependência de dados no codificador dificulta o processamento múltiplo de compartimentos em paralelo.

As técnicas utilizadas para melhorar a eficiência do CABAC apresentadas no trabalho, incluem:

- Redução dos *bins* codificados pelo contexto: os *bins* codificados pelo *bypass* podem ser codificados mais rapidamente que os *bins* codificados pelo contexto, pelo fato de

não possuírem dependência de dados, por conta da seleção de contexto. Com isso, sua divisão pode ser realizada por um simples deslocamento de bits. Na Tabela 5 é possível notar que os *bins* codificados pelo contexto são menores no HEVC do que no H.264. Na Tabela 6 também é possível analisar que no HEVC possui 8x menos *bins* codificados por contexto que no H.264, entre outros dados importantes que demonstram a melhora do CABAC no padrão HEVC.

Tabela 5 – Distribuição dos *bins* codificados por contexto, *bypass* e *terminate*.

	Configurações de condições comuns	contexto (%)	*bypass* (%)	*terminate* (%)
H.264	HierB	80.5	13.6	5.9
	HierP	79.4	12.2	8.4
HEVC	AI_MAIN	67.9	32.0	0.1
	LP_MAIN	78.2	20.8	1.0
	LB_MAIN	78.2	20.8	1.0
	RA_MAIN	73.0	26.4	0.6
	AI_HE10	68.1	31.8	0.1
	LP_HE10	78.6	20.4	1.0
	LB_HE10	78.6	20.4	1.0
	RA_HE10	73.3	26.1	0.6

Fonte: modificado de (SZE; BUDAGAVI, 2012)

Tabela 6 – Pior caso e redução de memória no HEVC e H.264.

Metric	H.264	HEVC	*Reduction*
Max context coded bins	7805	939	8x
Max bypass bins	13056	13425	1x
Max total bins	20861	14364	1.5x
Number of contexts	447	154	3x
Line buffer for 4K x 2K	30720	1536	20x
Coefficient storage	8x8x9-b	4x4x4-b	9x
Initialization table	64032	3536	18x

Fonte: modificado de (SZE; BUDAGAVI, 2012)

- Agrupar os *bins* de *bypass* para aumentar o número de *bins* processados em um ciclo, resultando assim em um menor número de ciclos necessários para o processamento dos *bins* de *bypass*. Com esse agrupamento o HEVC possui uma redução de 2,8x o número de ciclos comparado ao não agrupamento dos mesmos.

- Agrupar os *bins* com o mesmo contexto é outra técnica utilizada, tendo como resultado um número menor de cálculos especulativos que são utilizados para decodificar vários *bins* por ciclo, pois todos os *bins* que utilizam as mesmas regras para seleção de contexto são agrupados.

- Reduzir as dependências de seleção de contexto, para obter ganho na codificação sem resultar em penalidades significativas na taxa de transferência.

- Reduzir o número total de *bins*. Em uma comparação, o HEVC possui um número de 1.5x menor que o H.264. Supondo que a quantidade de ciclos por *bin* seja igual, o HEVC possui um tempo de 1.5x menor com uma economia de energia de 50%, podendo processar a uma taxa de 1.5x mais rápido que o H.264.

- Reduzir ou remover as dependências de análise para se ter um ganho de codificação sem ter uma penalidade significativa na taxa de transferência. Remover a dependência de análise para mesclagem e predição do vetor de movimento, permite que a análise seja desacoplada de outros módulos de vídeo, como no H.264. No padrão HEVC, existe dependência na reconstrução *Intra Mode* que não possui no H.264, porém, fizeram de tudo para manter a reconstrução do *Intra Mode* simples para evitar prejudicar o processamento da análise.

Outra mudança importante no HEVC foi a redução dos requisitos de memória, como por exemplo, a redução de contexto. O HEVC utiliza apenas 154 contextos, enquanto o H.264 utiliza 447 (ou 298), como é mostrado na Tabela 7. Com isso, é reduzido 3x o tamanho da memória do contexto. Outras reduções de memória estão descritas no trabalho de (SZE; BUDAGAVI, 2012), na qual todas contribuíram para melhorar a eficiência do HEVC.

Tabela 7 – Requisitos de memória de contexto do H.264 e HEVC.

	H.264		HEVC
	(w/ interlace)	(w/o interlace)	
CU *contexts*	23	20	14
PU *contexts*	26	26	14
TU *contexts*	398	252	124
LF *contexts*	n/a	n/a	2
Total	**447**	**298**	**154**

Fonte: modificado de (SZE; BUDAGAVI, 2012)

3.3 Algoritmos e Desenvolvimento de Arquitetura para a Codificação Binária Adaptativa ao Contexto para o Decodificador H.264

A dissertação de (DEPRA, 2009) tem como objetivo principal o desenvolvimento de uma arquitetura de *hardware* para o decodificador de entropia, que é executado especificadamente pelo CABAC, do padrão H.264. O trabalho também busca objetivos secundários, por conta da complexidade do algoritmo de decodificação do CABAC. As tarefas incluem a investigação detalhada do funcionamento dos algoritmos, a análise das

restrições do desempenho dos algoritmos e a análise do comportamento dinâmico do *bits-tream* gerado pelo processo de codificação do padrão H.264, explorando a possibilidade de novas arquiteturas.

Para a realização do trabalho, foi utilizado o software de referência JM 10.2 do padrão H.264. O software sofreu alterações para possibilitar posteriormente as avaliações dos resultados. Entre as informações resultantes do estudo dos algoritmos do CABAC, (DEPRA, 2009) destacou-se duas observações importantes:

- A existência de uma forte dependência de dados que tornam o CABAC e o CABAD (decodificador aritmético adaptativo ao contexto) em um gargalo para os compressores de vídeo (DEPRA, 2009).

- Não é possível determinar o número de bits produzidos durante o processo de decodificação para cada tipo de elemento sintático codificado, dificultando determinar a vazão necessária para o pior caso (DEPRA, 2009).

A conclusão do trabalho, referente a arquitetura desenvolvida, mostra que a mesma atingiu um desempenho suficiente para decodificar dados de vídeos na resolução HD1080 em tempo real. O desempenho máximo obtido pela arquitetura alcança uma frequência de 99 MHz, porém, no caso médio, vídeos com resolução HD1080, opera em uma frequência aproximada de 31 MHz em tempo real.

4 Método Proposto

O método proposto neste trabalho é baseado nas observações dos trabalhos propostos por (ROSA, 2010) e (DEPRA, 2009). A partir destas observações, alterações foram realizadas nos software de referência dos padrões de codificação de vídeo H.264 e HEVC. Estas alterações estratégicas permitiram observar o comportamento do *bitstream* do vídeo codificado e a adoção de simplificação de elementos sintáticos resultaram em análise dos resultados da codificação. Com a adoção desta estratégia, foi possível observar o quanto o agrupamento de elementos sintáticos impacta na codificação em termos de tamanho da saída, qualidade visual e complexidade computacional (tempo de codificação).

Para realizar a codificação dos vídeos foram utilizados dois softwares de referência: para o padrão H.264/AVC foi utilizado como base o software de referência JM (*Joint Test Model*) (ITU-T ISO/IEC, 2018a) na versão (JM-18.0) e para o padrão HEVC o software de referência utilizado foi o HM (HEVC *Test Model*) (ITU-T ISO/IEC, 2018b) na versão (HM-16.19).

Após a realização do download dos softwares de referência e também as compilações dos mesmos seguindo o manual do JM (TOURAPIS et al., 2009) e do HM (BOSSEN; FLYNN; SUHRING, 2013), foram escolhidos vídeos para compor a base de dados que seriam utilizados nas experiências. Os vídeos escolhidos possuiam diferentes resoluções, tamanhos e formatos, para uma melhor análise dos dados. Previamente, todas as codificações foram realizadas antes de sofrerem alterações utilizando o software original nas versões supracitadas. Foram codificados 80 frames de cada sequência de vídeo mantendo um mesmo padrão de saída para todos os vídeos. Cada codificação resultou um arquivo YUV (reconstruído), um hevc resultante do HM e um h264 resultante do JM. Os softwares de referência utilizados apresentam *feedback* com algumas estatísticas em relação a codificação, no caso do JM, um arquivo de estatística. Já o HM mostra as estatísticas gerais de codificação no próprio terminal a qual foram transferidas para um arquivo para uma melhor avaliação das mesmas.

As configurações do computador utilizado para realizar as experiências utilizando os softwares de referências estão especificadas na Tabela 8.

Tabela 8 – Configuração do hardware utilizado.

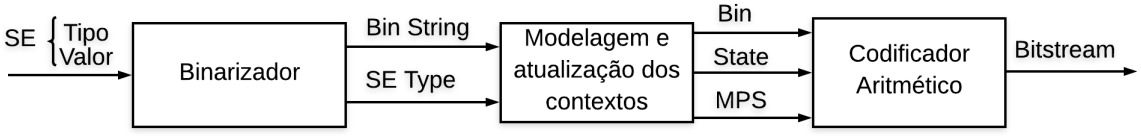
Modelo	Microcomputador Portatil Dell Inspiron 3421
Memória	6 GB
Disco	1TB
Tipo de Sistema	64 bits
Gráficos	Intel Ivybridge Mobile
Processador	CPU Intel Core i5-3337U
Número de Núcleos	2
Número de threads	4
Frequência Baseada em Processador	1,80GHz
Frequência Turbo Max	2,70GHz
Cache	3MB SmartCache
Velocidade do barramento	5GT/s DMI
TDP	17W

Além das extrações de estatísticas de codificação dos vídeos no software de referência, também foram realizadas codificações com as alterações no software para medir o impacto de simplificações no processo de codificação aritmética. Estas simplificações tem sempre o objetivo de reduzir o gargalo de paralelismo do processo. Na próxima seção, será apresentado detalhadamente o processo de codificação aritmética binária adaptativa ao contexto.

4.1 Visão Geral do CABAC

O codificador aritmético binário adaptativo ao contexto é um codificador de entropia que tem como objetivo transformar valores de um símbolo em uma palavra de código com tamanho variável. Combinando divisão recursiva de intervalo e modelo de contexto o CABAC se torna mais eficiente em relação a codificação. Um subintervalo representa um símbolo e o tamanho do intervalo está relacionado a probabilidade de ocorrência desse símbolo. Na Figura 12 é possível visualizar o fluxo básico de dados do CABAC.

Figura 12 – Fluxo de dados no CABAC.



Fonte: modificado de (ROSA, 2010)

A entrada do codificador de entropia são os elementos sintáticos resultantes dos outros blocos do codificador, incluindo informações de bloco, vetores de movimento e resíduos quantizados. Todo elemento sintático possui uma identificação (um tipo) e também um valor. Como o CABAC só codifica valores binários, todos aqueles elementos

sintáticos não binários são binarizados. O processo de binarização tem como saída uma string de tamanho variável de valores binários, os quais são chamados de *bins*. Contextos relacionados ao tipo de elemento sintático e a posição do *bin* na string que formam o elemento sintático binarizado são utilizados como informações para selecionar o contexto que será utilizado para a codificação daquele *bin*. Por isso, *bins* diferentes que pertencem a um mesmo elemento sintático podem ter informações de contexto diferentes. Após a binarização, o codificador binário aritmético tem como entrada o *bin* e a informação de contexto. Posteriormente, o modelo de contexto é atualizado e o codificador tem como saída zero ou mais bits.

4.1.1 Processo de Binarização

A binarização tem como objetivo transformar os valores dos elementos sintáticos em uma única sequência de bits na qual representam o valor original. Esse processo serve para diminuir os símbolos do alfabeto que serão codificados e modelados, reduzindo custos em relação a modelagem e também auxiliando o codificador aritmético nas suas tarefas.

No processo de binarização cada bit é chamado de *bin* e um conjunto de todos os *bins* é denominado *binstring* (string de *bins*).

4.1.2 Modelagem de Probabilidade

Um modelo de contexto tem como objetivo identificar a probabilidade de ocorrência do símbolo que será codificado. O modelo de contexto é um modelo com probabilidades que apresenta as estatísticas de um símbolo tendo como referência a ocorrência daqueles símbolos que já foram codificados. Os *bins* dos elementos sintáticos podem estar relacionados a um ou mais contextos, individualmente. Na codificação é preciso manter as probabilidades estatísticas atualizadas para que o processo seja eficaz.

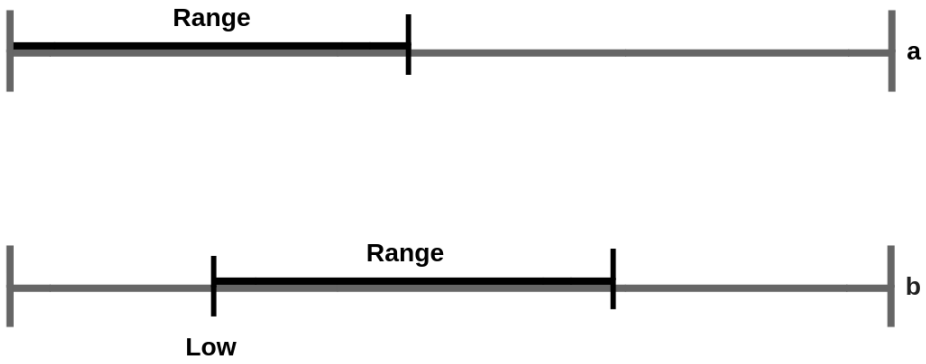
4.1.3 Codificador Aritmético Binário

O codificador aritmético binário realiza a codificação com base no princípio da subdivisão recursiva de intervalo na qual o seu tamanho é representado pela letra R. Após ser encontrado a probabilidade do símbolo menos provável (LPS) em uma determinada faixa, são gerados dois subintervalos. Um é obtido por rLPS = R*pLPS, relacionado com o LPS, já o segundo é obtido por rMPS = R-rLPS, relacionado ao MPS. Dependendo do *bin* codificado o rLPS ou o rMPS será o novo intervalo de R. Após esse processo, o R é quantizado e multiplicado por rLPS, o resultado é armazenado em uma tabela fixa de 64x4 indexada pelos 6 bits de estado do modelo de contexto e pelos bits 6 e 7 do R. Essas operações são realizadas para diminuir a complexidade computacional. Durante a codificação aritmética binária, o registrador de faixa R (Range) e o de deslocamento "O"

(Offset) precisam ser atualizados. O R armazena o valor do intervalo atual enquanto o "O" marca o limite inferior dentro deste intervalo.

Um intervalo unitário no H.264 possui o tamanho de 1024 valores discretos possíveis na qual possui a informação codificada. Um exemplo da condição inicial do intervalo e o após alguns passos da codificação aritmética pode ser visto na Figura 13.

Figura 13 – Intervalo inicial (a) e após vários passos de codificação (b).



Fonte: modificado de (ROSA, 2010)

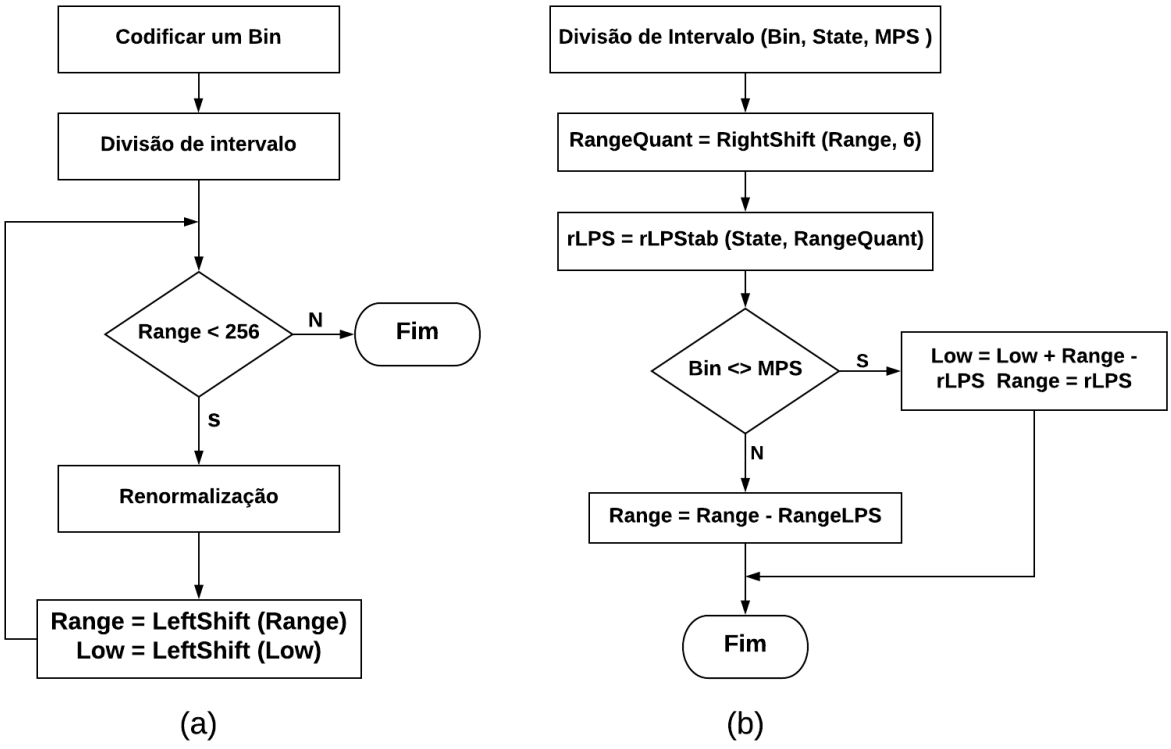
O intervalo unitário está dividido em 1024 posições de 0 a 1023. O valor do intervalo R deve estar sempre entre 256 e 511 e o valor da base do intervalo (L) entre 0 e 511. Para que o intervalo permaneça no mesmo tamanho, após alguns passos de codificação, é necessário que o codificador passe pelo processo de renormalização a cada vez que um *bin* for codificado, como mostra a Figura 13.

Toda vez que um *bin* é codificado, independente da sua probabilidade, irá ser reduzido o intervalo pelo processo de divisão do intervalo. Para que não seja reduzido o intervalo além do limite de resolução permitido a cada *bin* codificado, é realizado o processo de renormalização para que o tamanho do intervalo fique entre 256 e 511. Quanto maior for a probabilidade, menor será a redução do intervalo e o número de vezes que o processo de renormalização é ativado também será reduzido. A diminuição de intervalo sempre acontece, pois a probabilidade de ocorrência de um *bin* é menor que a unidade, enquanto a multiplicação do intervalo por um valor menor que a unidade vai gerar uma diminuição no seu tamanho. Toda vez que o processo de renormalização é executado, um bit é enviado para o *bitstream*. Por isso, quanto menos ser renormalizado melhor.

Para encerrar a codificação, um modo de codificação especial é executado. Esse é chamado de modo final (*terminate*). O modo *terminate* ocorre sempre que a codificação aritmética precisa ser finalizada. Na Figura 14 está representado o fluxograma de codificação aritmética para um *bin*.

O primeiro passo da codificação do *bin* é a divisão de intervalo que reduz o valor de R (Range) dependendo da probabilidade de ocorrência do valor do *bin*. No fluxograma (b), apresentado na Figura 14, representa o processo de divisão do intervalo. Após o *bin*

Figura 14 – Processo de codificação aritmética para um *bin* (a) e processo de divisão de intervalo (b).



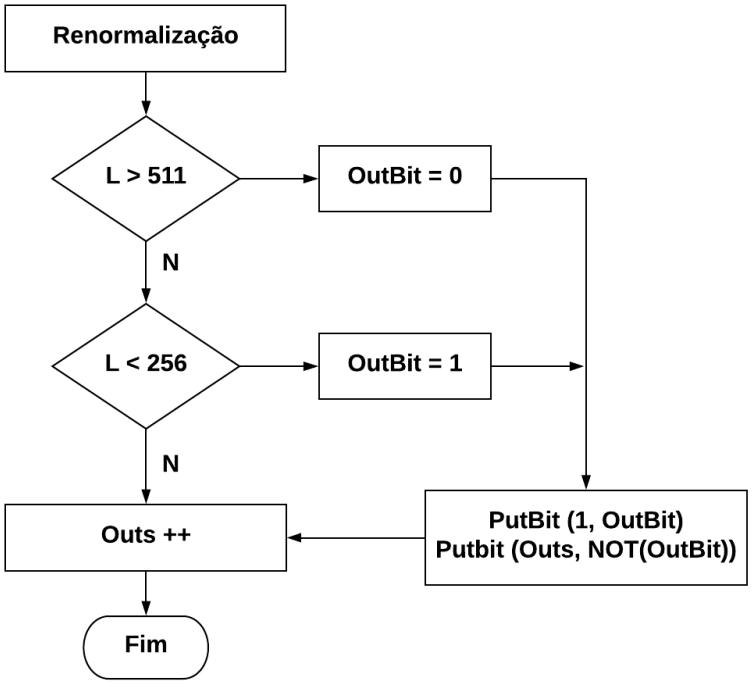
Fonte: modificado de (ROSA, 2010)

passar pelo processo de divisão de intervalo, ele segue para um loop de renormalização que só será encerrado quando o R se tornar maior ou igual a 256.

A cada renormalização executada um bit é enviado ao *bitstream*. O valor deste bit está relacionado a consulta ao registrador de base do intervalo L (Low). Se este registrador, que marca o início do intervalo, for maior que 511, um bit de valor 0 é enviado ao *bitstream*. Porém, se o valor de L for menor que 211, é enviado um bit de valor 1 ao *bitstream*. Toda vez que a renormalização é executada ocorrendo a produção de um bit, o intervalo é multiplicado por dois para que se evite a produção de um número infinitesimal para representar o intervalo. O envio de bits para o *bitstream* é a função mais importante de um número, pois representa um ponto do intervalo R que após a codificação será utilizado no processo de decodificação. Existe um caso em que o valor de L está entre 256 e 511, quando isso ocorre, é impossível decidir o valor do bit que vai ser enviado ao *bitstream*. A operação aritmética de criação do *bitstream* ocorre por meio de consecutivas adições de valores de 9 bits N passos de renormalização a direita gerando um grande número de dígitos intitulados *bitstream*. Durante a codificação, pode ocorrer o chamado "vai um", que modifica o valor de dígitos já calculados, ao qual está à esquerda do bit mais importante da operação que está sendo realizada (9 bits), precisando de uma

atualização desses valores. Esses bits são chamados de *outstading bits*. É utilizado um contador para marcar todos aqueles bits *outstading* que são produzidos durante o processo de renormalização, na qual pode ser visualizado na Figura 15. Toda vez que um bit 0 ou 1 é gerado, a condição "vai um" é resolvida, e todos aqueles bits que dependem da decisão, que são marcados como *outstading*, são enviados ao *bitstream*. Isto é resolvido pela chamada do processo "PutBit", na qual pode ser visualizado na Figura 15. Na primeira parte ele envia os *outstading* já resolvidos e na segunda parte envia o bit codificado que gerou a decisão (ROSA, 2010).

Figura 15 – Processo de renormalização.

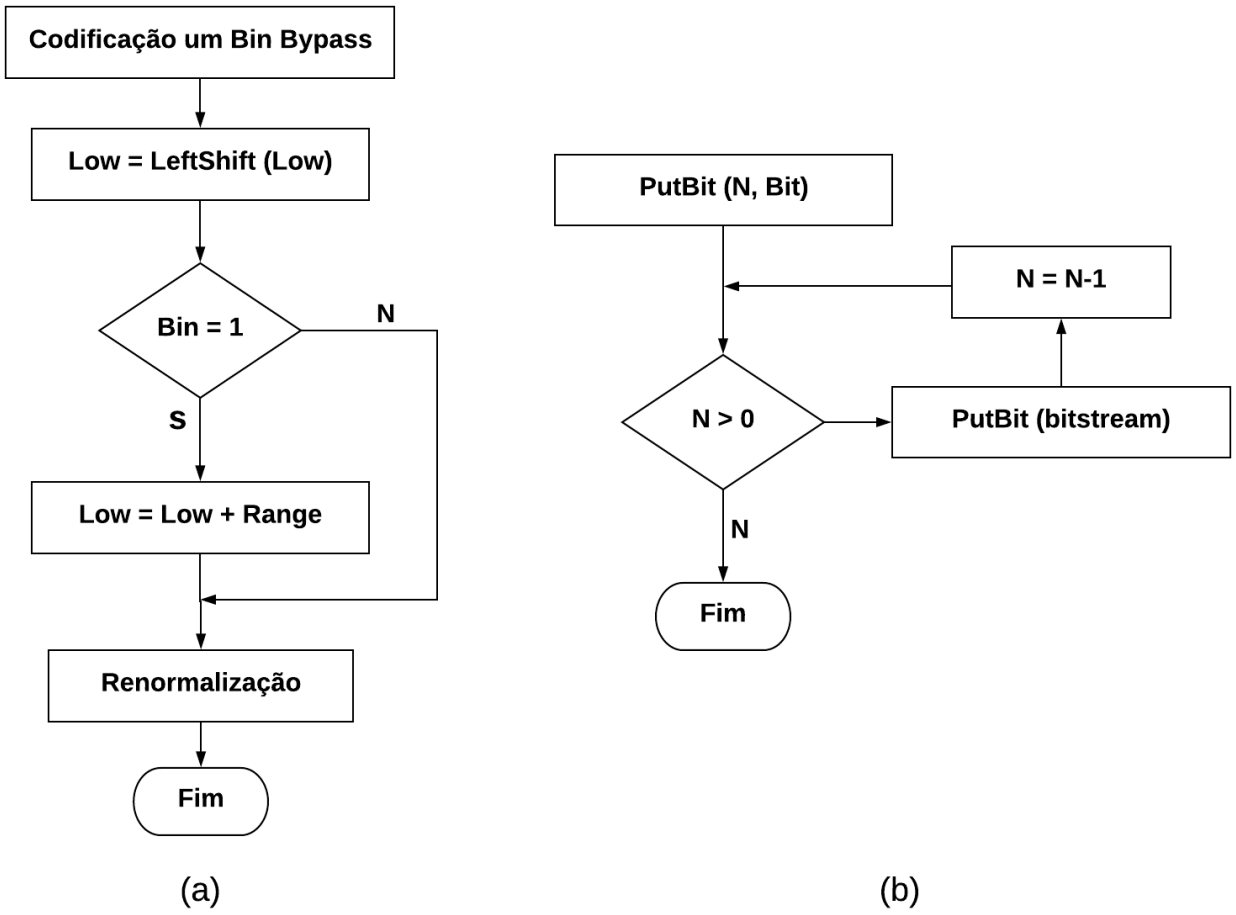


Fonte: modificado de (ROSA, 2010)

Como já foi explicado anteriormente, os *bins* que possuem probabilidade próxima a 0,5 (resultado determinado pelo padrão) é utilizado o método de codificação *Bypass*, na qual é representado no fluxograma (a) da Figura 16. Já o fluxograma (b) da Figura 16 representa o algoritmo de inserção de bits no *bitstream*.

Após de extração de informações e análises dos dados obtidos das codificações de vídeo, segundo (DEPRA, 2009) em relação ao codificador de vídeo H.264, a taxa de compressão alcançada pela codificação aritmética é de 1,3 *bins* processados para cada bit produzido, porém, em algumas sequências podendo chegar a 2,8 *bins* por bit. Por fim, pode-se concluir que a maior parte da compressão ocorrerá ao decorrer do processo de binarização e a capacidade de compressão do codificador aritmético possui uma forte dependência do processo de modelagem de contextos empregada (ROSA, 2010).

Figura 16 – Processo de codificação *Bypass* (a) e processo de inserção de bits no *bitstream* (b).

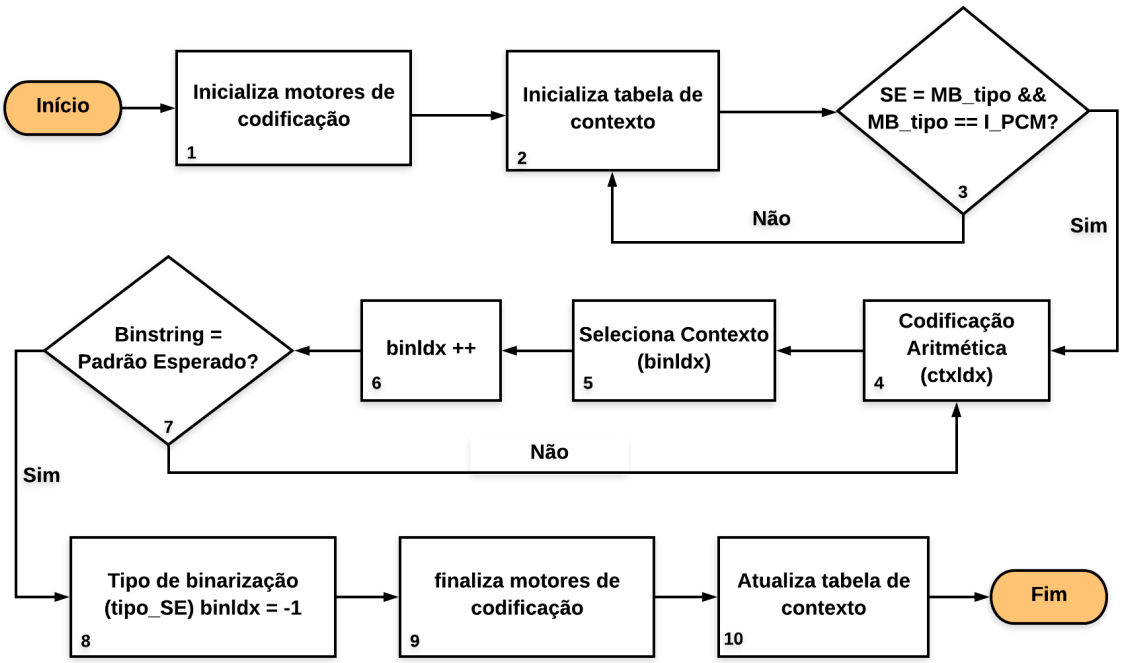


Fonte: modificado de (ROSA, 2010)

4.2 Características do *Bitstream*

Para melhor análise do comportamento dos elementos sintáticos foi necessário estudar as características do *bitstream* e também obter conhecimento acerca do fluxo de execução do CABAC. Na Figura 17 é apresentado o diagrama de fluxo de execução do CABAC. Todos os elementos sintáticos (SE) passam por esse processo de codificação até que o SE que marca o final do bloco seja encontrado. Esse processo de codificação que possui algumas etapas pode também ser dividido em três partes: inicialização, codificação e encerramento.

Figura 17 – Diagrama de fluxo de execução do CABAC.



O primeiro passo do fluxo é a inicialização dos motores de codificação, após esta etapa a inicialização da tabela de contexto é acionada. Em seguida o elemento sintático passa por um teste condicional na qual dependendo do resultado é selecionado uma nova etapa. Então se a condição for satisfeita a codificação aritmética será a próxima. Se a condição da etapa for falsa retorna na tabela de contexto que é atualizada. As etapas 4, 5, 6 e 7 representam o laço de repetição que codifica um *bin* por inserção até que todos os *bins* que representam o valor do SE sejam codificados em uma *binstring*. A etapa 8 tem como função identificar o tipo de binarização esperada para o SE que vai ser codificado e inicializa o contador de *bins* codificados do SE atual. Na etapa 9 é finalizado os motores de codificação. Por fim, é atualizado a tabela de contexto antes do término do fluxo.

4.3 Alterações nos Softwares de Referência

O software de referência fornece a estatística geral dos elementos sintáticos e suas dependências, entretanto, não fornece quais os elementos sintáticos que são responsáveis por gerarem dependências entre si. Desta forma, é necessário realizar o mapeamento daqueles elementos sintáticos que não geram dependências e que podem ser paralelizados. Conforme (ROSA, 2010), o limite de produção de paralelismo para o H.264/AVC é de 30%, devido a estas dependências. Já, para a análise do HEVC, não foram encontrados registros de estudos de dependência dos elementos sintáticos produzidos pelo CABAC, sendo considerada paralelismo em nível de CTU.

As alterações foram realizadas nos softwares e em seus arquivos de configurações. Os arquivos de configuração possuem operações básicas como também operações mais

avançadas que auxiliam na codificação e na extração de informações acerca da compressão de vídeos. Dentre as muitas funções dos arquivos de configuração, serão destacadas abaixo os comandos mais relevantes e utilizados no arquivo de configuração e suas funcionalidades:

- ***File:*** refere-se aos arquivos de entrada e de saída da codificação, na qual está contido: a especificação do arquivo de entrada; o nome do arquivo de saída desejado; quantos frames por segundo possui o vídeo (FPS); quantos bits de cores o vídeo a ser codificado possui; a resolução do vídeo; e quantos frames o usuário quer codificar variando de 1 frame até o número máximo de frames que o vídeo possui. Todas essas informações devem ser especificadas corretamente para que seja possível realizar a codificação.

- ***Unit definition:*** podendo ser definido a largura máxima da unidade de codificação em *pixels*; a altura máxima da unidade de codificação em *pixels*; a profundidade máxima da unidade de codificação; entre outras.

- ***Motion Search:*** na pesquisa de movimento é possível definir a faixa de pesquisa de movimento adaptativo; o tamanho mínimo da janela de pesquisa de movimento para a janela adaptativa ME (estimação de movimento); se a pesquisa vai ser completa ou por zona de teste (TZ search); o uso da medida de Hadamard para ME fracional; a decisão rápida de codificador; a decisão rápida para o custo de mesclagem RD (taxa-distorção); entre outras definições possíveis.

- ***Quantization:*** é possível definir no arquivo de configuração as especificações relacionadas a quantização, podendo ser definido: parâmetro de quantização (QP) que varia de 0 a 51; otimização de parâmetros de quantização múltipla baseada em unidade de codificação; otimização de parâmetros de quantização múltipla baseada em fatias; quantização otimizada de distorção de taxa; Quantização otimizada de distorção de taxa para transformada skip; entre outras funcionalidades.

- ***Coding Structure:*** defini-se que o acesso aleatório pode ser nulo, *clean random access*, IDR ou ponto de recuperação de informações; tamanho do GOP; escrita ou não de conjuntos de parâmetros com cada *intra random access pictures*; entre outras especificações.

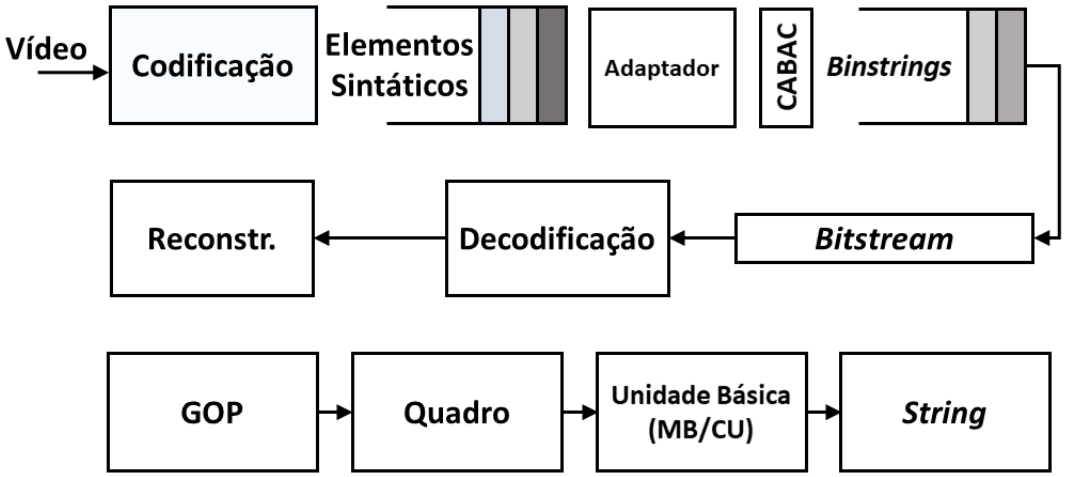
- ***Slices:*** aplicar o número máximo de LCU em um *slice*; aplicar o número máximo de bytes em um *slice*; desativar todas as opções de *slices*; entre outras definições.

- ***Coding Tools:*** ativação ou não do deslocamento adaptativo de amostra; das partições de movimentos assimétricos; pular o processo de transformada; pular processo de transformada antecipadamente.

- **PCM:** defini o Log2 do tamanho máximo do bloco PCM; defini o Log2 do tamanho mínimo do bloco PCM; defini se a profundidade do bit PCM deve ser a profundidade de bits interna ou a profundidade do bit de entrada; ativa ou desativa a filtragem de loop em amostras I_PCM.
- ***Rate Control:*** ativar ou não o controle de taxa; selecionar entre alocação de bit igual, alocação de bit de taxa fixa ou relação adaptativa; nível de LCU RC ou nível de imagem RC; forçar intra QP para ser igual a QP inicial.
- ***Lossless:*** pode-se definir o valor de sinalização; e também forçar ou não o modo de desvio de *transquant*, quando *transquant_bypass_enable_flag* estiver ativado.
- ***Quantization Matrix:*** possui as opções de estar ativo ou não; é possível especificar o nome de saída do arquivo txt.
- ***Tiles:*** define algumas configurações relacionadas a linhas e colunas como, escolher entre os limites de coluna e linha a serem distribuídos uniformemente ou os limites da coluna a serem indicados por uma matriz X e os limites da linha a serem indicados por uma matriz Y; entre outras configurações.

A extração dos resultados através dos softwares de referência está em conformidade com a metodologia proposta por (DEPRA, 2009). As diferentes características apresentadas pelos codificadores de cada padrão que foram apresentadas no decorrer deste texto requereram ajustes específicos em cada uma das versões. Os elementos sintáticos produzidos pelos blocos codificadores de cada padrão são distintos, gerando estatísticas diferente entre os agrupamentos de SEs. Mesmo assim, o esquema de verificação de sequências de elementos sintáticos agrupados é o mesmo para os dois padrões, sendo apresentado pela Figura 18.

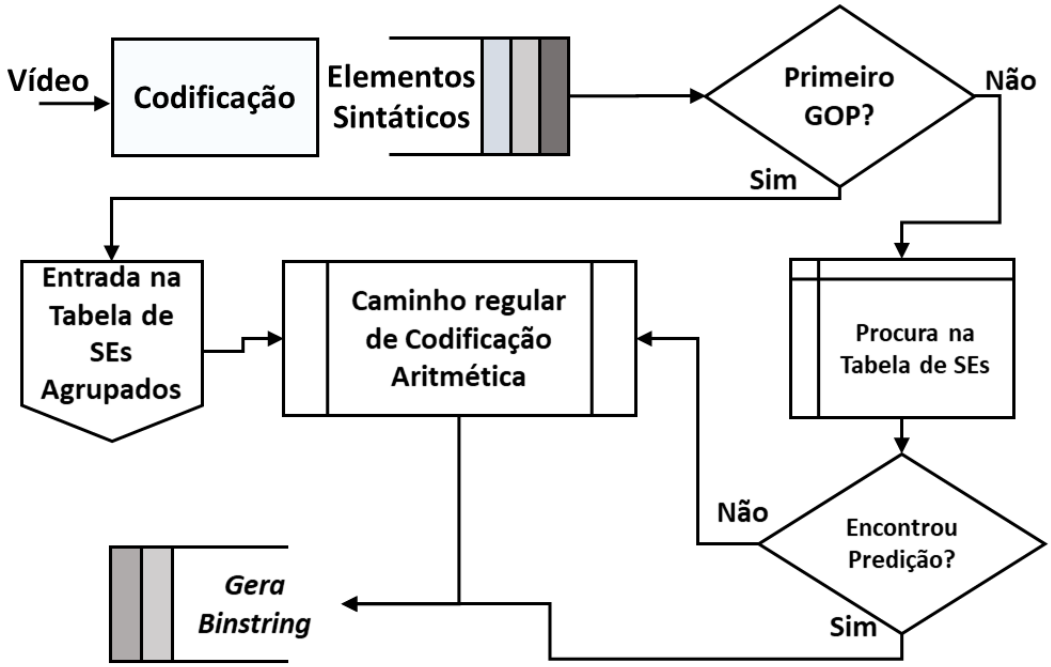
Figura 18 – Diagrama de fluxo do algoritmo de predição de elementos sintáticos.



Durante todo caminho de codificação são gerados elementos sintáticos que serão transformados em símbolos no CABAC. Antes da codificação aritmética, entretanto, foi inserido um novo módulo nos softwares de referência, chamado adaptador. Esse módulo avalia a ocorrência de símbolos agrupados através do caminho de codificação GOP, quadro e unidade básica até a finalização do CABAC para composição de uma string que gera o *binstring*, que por sua vez gera o *bitstream* a ser armazenado ou transmitido e posteriormente decodificado.

O módulo adaptador, apresentado na Figura 19, faz uma rotina para construir a tabela de elementos preditos a partir do primeiro GOP de cada sequência de amostras. Os elementos sintáticos que possuírem ocorrências agrupadas, não precisarão passar por codificação. No decorrer da codificação dos demais GOPs do vídeo, o algoritmo consulta esta tabela para verificar se o elemento gerado está presente, em caso positivo, gera o *binstring* sem passar pelo caminho de codificação do CABAC, senão encontrar, passa pelo caminho regular de codificação. Desta forma, este adaptador atua de forma análoga ao módulo *bypass* em um nível mais alto.

Figura 19 – Diagrama de fluxo do algoritmo de predição de elementos sintáticos.



4.4 Analisador de Fluxo de Bits Gitl HEVC

Para auxiliar nas análises do fluxo de bits da codificação do HEVC foi utilizado um analisador chamado Gitl HEVC (LI, 2018). O analisador Gitl HEVC é uma ferramenta de código aberto com a função de analisar o fluxo de bits para o HEVC. O analisador possui várias funções que podem ser selecionadas juntas ou individualmente sendo possível visualizá-las quadro por quadro. Dentre as principais funções estão: o tipo de predição

que está sendo utilizada; a exibição da Unidade de Codificação (CU), da Unidade de Predição (PU), da Unidade de Transformação (TU), dos Vetores de Movimento (MV) e a exibição *Intra-Mode* (Angular, DC, Planar). Na Figura 20 é mostrado a tela do analisador com a função *Intra Mode Display* selecionada.

Figura 20 – Tela do analisador de fluxo de bits Gitl HEVC.

5 Resultados

Neste capítulo serão apresentados os resultados obtidos através da codificação nos softwares de referência. Os resultados foram gerados a partir da codificação aplicada em uma base de amostras de vídeos puros no formato YUV. Também serão apresentados os resultados da avaliação dos softwares de referência analisando a possibilidade de utilização de técnicas de paralelismo no processo de codificação aritmética. Por fim, uma breve comparação entre as codificações feitas nos softwares de referência do H.264/AVC e HEVC.

5.1 Amostras

Para o desenvolvimento do presente trabalho foram utilizados vídeos no formato YUV selecionados a partir do documento de referência dos softwares *Common Test Condition* (BOSSEN et al., 2013) para a codificação e posterior avaliação dos resultados obtidos. A Tabela 9 apresenta a sequência de todos os vídeos utilizados durante o processo de desenvolvimento do trabalho no qual os vídeos estão categorizados entre as resoluções VGA, XGA, HD720, HD1080 e Ultra HD. A seleção dos vídeos abrange as três principais características apresentadas como condições pelas referências: a) *Intra Only*, b) *Low Delay* e c) *Random Access*. Como pode ser observado na Tabela 9, nem todos os vídeos foram codificados pelo software de referência JM, isso ocorre porque o mesmo não suporta resoluções superiores a FullHD (1080p). As resoluções menores ou iguais que não foram codificadas, foram devido à falta de tempo para realização das operações.

Tabela 9 – Relação das sequências de vídeo.

vídeo	Resolução	Quadros	Utilização
Race	VGA (640x480)	300	JM / HM
Exit	VGA(640x480)	250	JM/HM
Flamenco	VGA (640x480)	300	JM / HM
Uli	XGA(1024x768)	250	JM/HM
Ducks	HD720 (1280X720)	500	JM / HM
Park	HD720(1280X720)	500	JM/HM
Beauty	HD1080 (1920x1080)	600	HM
Bosphorus	HD1080(1920x1080)	600	HM
HoneyBee	HD1080 (1920x1080)	600	HM
ReadySetGo	HD1080(1920x1080)	600	HM
ShakeNDry	Ultra HD (3840x2160)	300	HM
Jockey	Ultra HD (3840x2160)	600	HM

As imagens apresentadas a seguir representam todas as amostras de vídeo utilizadas no presente trabalho que foram apresentadas na Tabela 9.

Figura 21 – Sequências: Jockey, Beauty, HoneyBee.

Figura 22 – Sequências: Bosphorus, ReadySetGo, ShakeNDry.

Figura 23 – Sequências: Park, Uli, Ducks.

Figura 24 – Sequências: Race, Flamenco, Exit.

A partir das próximas linhas, veremos os resultados comparativos obtidos pelos codificadores genéricos dados pelos software de referência e pela suas versões alteradas com o agrupamento de elementos sintáticos codificados em conjunto.

5.2 Resultados Gerais

A Tabela 10 apresenta os resultados obtidos pela compressão pura dos dois softwares de referência sobre as amostras selecionadas. Os resultados obtidos pelo software do HEVC são sempre superiores aos obtidos pelo H.264 conforme o esperado. As amostras

de vídeo com resoluções maiores que 1920x1080 não podem ser codificadas pelo padrão H.264, desta forma, não há resultado para estas neste padrão. Finalmente, as amostras na resolução 1920x1080 não foram codificadas no padrão H.264 devido a alta demanda de tempo por codificação de amostra na escolha do algoritmo *Full Search*. Novas codificações com outros algoritmos para escolha de referências na geração de vetores de movimento poderão ser feitas no futuro.

Tabela 10 – Comparação das taxas de compressão HEVC vs H.264.

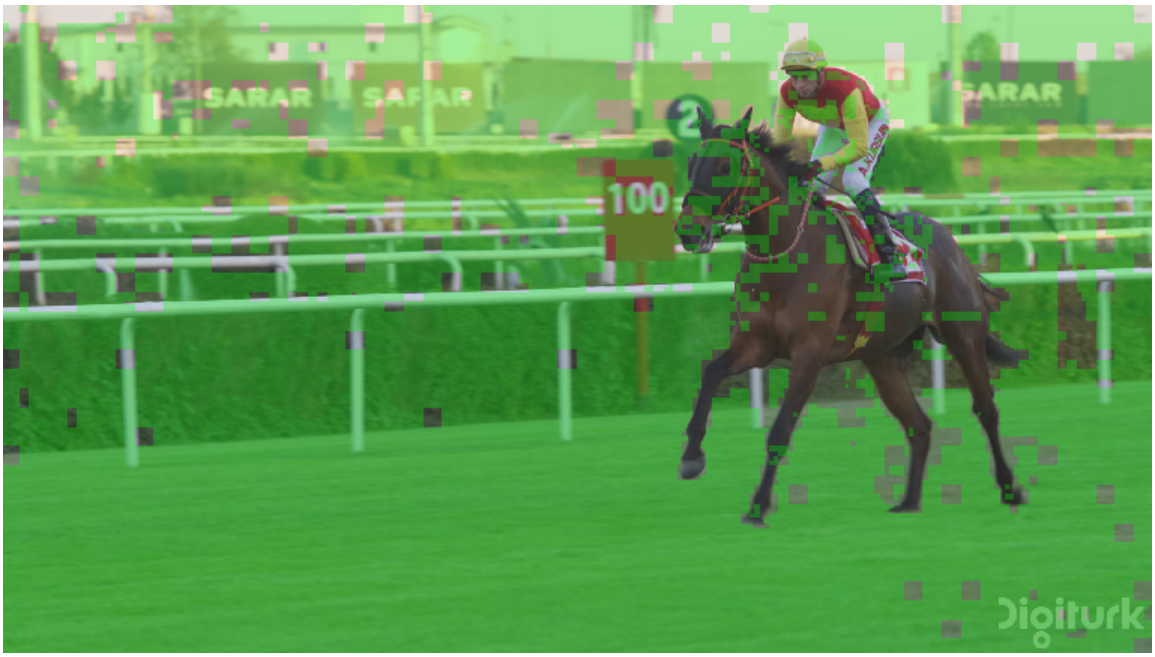
Sequências	Comparação entre Taxas de Compressão						
	Entrada (MB)	HEVC			H.264		
		Saída (KB)	Redução (%)	PSNR (dB)	Saída (KB)	Redução (%)	PSNR (dB)
Race	36	128	0,356	32,61	379	1,053	32,65
Exit	36	71	0,197	33,8	160	0,444	33,86
Flamenco	36	173	0,481	34,71	373	1,036	34,84
Uli	90	356	0,396	35,57	667	0,741	35,68
Ducks	105	798	0,76	38,6	2080	1,981	38,74
Park	105	1150	1,095	39,56	2780	2,648	39,75
Beauty	237	172	0,073	40,34	-	-	40,46
Bosphorus	237	204	0,086	40,88	-	-	41,03
HoneyBee	237	167	0,07	35,25	-	-	35,66
ReadySetGo	237	452	0,191	37,04	-	-	37,34
ShakeNDry	949	1840	0,194	38,55	-	-	38,84
Jockey	949	465	0,049	39,96	-	-	40,24
Média	271	498	0,184	37,24	1073	1,317	37,42
Maior Valor	949	1840	-	40,88	-	-	41,03
Menor Valor	36	71	-	32,61	-	-	32,65

A Figura 25 mostra a diferença das *Coding Units* do padrão HEVC para os antigos macroblocos, a adoção destas estruturas impactou diretamente nos elementos sintáticos criados. Estes elementos sintáticos poderão ser majoritariamente agrupados de acordo com a ordem de codificação e *binstring* gerado. Reforçando o que já foi apresentaado na seção 2.2 o padrão HEVC possui a divisão de blocos mais flexível, que podem ter o tamanho máximo de 64x64 e serem divididos em vários tamanhos. Já no padrão H.264, o tamanho máximo dos macroblocos é de 16x16. No HEVC as CTUs são divididas em CUs, que podem ser divididas por blocos de tamanho 64x64 chegando a partições de tamanho 8x8. As CUs podem ser particionadas em PUs que não precisam ter o formato quadrado, podem ser particionadas no formato retangular. As PUs podem ser grandes como as CUs ou pequenas no tamanho 8x4 ou 4x8. Já os macroblocos do padrão H.264 podem ter o tamanho máximo de 16x16 e a menor partição no tamanho de 4x4.

Figura 25 – Unidades de codificação e vetores de movimento.

A Figura 26 mostra os modos de predição “merge”, análogo ao elemento sintático do modo *skip* presente no H.264, ou seja, são as CUs que não vão gerar novas *binstrings* na codificação do presente quadro.

Figura 26 – Modos de predição merge.



A Figura 27 mostra os tipos de modo Intra adotados pelo codificador HM na sequência *Jockey*, os modos são relativos as texturas internas da imagem. Destacado, estão visíveis mais de uma dezena de modos aplicados, uma complexidade não existente no codificador do padrão anterior, como já foi visto na seção 2.2. O módulo de predição intra do padrão HEVC possui 35 modos de predição, sendo 33 angular e mais as predições DC e planar. Já o módulo de predição intra do padrão H.264, possui 9 módulos de predição para blocos 4x4 e 4 módulos de predição para blocos 16x6. Todos esses modos acabam gerando elementos sintáticos com comportamentos a serem analisados.

Figura 27 – Exemplo de modos intra no padrão HEVC.

5.3 Observações sobre o *Bitstream*

Na Tabela 11 é apresentado o tipo de elemento sintático (SE) e o significado da nomenclatura adotada. Em relação a Tabela 11, segundo (DEPRA, 2009) os coeficientes de transformadas quantizados foram agrupados apenas pelo tipo de componente (crominância/luminância) em vez de serem separados por conta do tipo de nível e do tipo de predição que foram gerados (Intra/Inter). Nos vetores de movimento diferencial (MVD) foi escolhido como opção agrupar os componentes XeY em um mesmo elemento, exceto entre as listas 0 e 1. Os índices para quadros de referência foram agrupados em um único SE independente da lista (DEPRA, 2009).

Tabela 11 – Nomenclatura adotada para os diversos tipos de SEs.

Tipo de SE	**Significado da Nomenclatura Adotada**
MBTYPE	Tipo de macrobloco
REFIDX	Índices para os frames de referência (Lista 0 e Lista 1)
CBP	Padrão de codificação dos blocos 8x8 (4 blocos para luminância e 2 blocos para crominância)
COEF_LUM	Coeficientes de transformada quantizados gerados pelos resíduos da predição (amostras de luminância)
COEF_CHR	Coeficientes de transformada quantizados gerados pelos resíduos da predição (amostras de crominância)
QP_DELTA	Variação do parâmetro de quantização no nível de macrobloco
COEF_SIG	Conjuntos de flags que compõem o mapa de significância (um flag para cada coeficiente de bloco 4x4)
COEF_LAS	Conjuntos de flags que compõem o mapa de significância (um flag para cada coeficiente significante)
CBF	Padrão de codificação dos blocos 4x4 dentro de uma partição 8X8 do macrobloco
INTRA_FLAG	Flag para indicar variação entre o modo de codificação do bloco Intra atual em relação ao anterior
INTRA_MODE	Modo de codificação do Intraquadros empregado pelo bloco de predição
SUB_MBTYPE	Tipo de sub-macrabloco, ou seja, forma como cada partição de um macrobloco é organizada
MVD_L0	Componentes X e Y para vetor de movimento diferencial relacionados a lista 0
MVD_L1	Componentes X e Y para vetor de movimento diferencial relacionados a lista 1

Fonte: modificado de (DEPRA, 2009)

Tendo uma produção de *bins* sequenciais do mesmo módulo de codificação aritmética incluído em um mesmo SE, abre caminho para que se possa explorar o espaço de projeto em relação a arquitetura com taxa de processamento variável, com a possibilidade de gerar mais de um *bin* por ciclo. Levando em consideração o tamanho médio dos SEs e também o tamanho máximo do *binstring* gerado por esses SEs, levanta-se a necessidade de analisar mais a fundo a utilização subsequente do mesmo módulo de codificação aritmética (DEPRA, 2009).

Segundo (DEPRA, 2009) para que fosse possível avaliar o impacto de incidências sequenciais em relação ao número de repetições acima de duas, se tornou essencial coletar informações adicionais. Por conta do grande volume de informações a análise se restringiu em apenas alguns SEs, na qual são dos tipos COEF_LUM, 108 COEF_CHR, MVD_L0 e MVD_L1 que fazem parte dos SEs que utilizam o módulo *Bypass* de codificação aritmética. Optou-se por analisar o módulo *Bypass* pelo fato de o módulo Regular apresentar mais complexidade e limitações, na qual dificultaria a utilização de uma solução arquite-

tural com diversos destes módulos aninhados (DEPRA, 2009).

Na Tabela 12 as informações apresentadas resumem o processo de análise acerca das ocorrências consecutivas do módulo *Bypass*. A Tabela 12 está organizada por resolução e tipo de SE analisado.

Tabela 12 – Análise da produção de *bins*.

Tipo de SE	BINS (%)	BYPASS			
		BINS no *Bypass* (%)	Consec. (%)	BINS Consec (%)	>2 *bins* Consec (%)
COEF_LUM	45,72	12,14	23,60	2,89	1,88
COEF_CHR	17,80	5,19	19,30	1,33	1,10
MVD_L0	0,92	0,24	47,50	0,11	0,10
MVD_L1	0,20	0,05	42,38	0,02	0,02
Sub-Total	64,65	17,64	-	4,37	3,10

Fonte: modificado de (DEPRA, 2009)

A coluna que possui o nome “BINS (%)” refere-se a taxa dos *bins* produzidos por cada SE relacionados ao total de *bins* gerados por todos os SEs. A coluna denominada “BINS no *Bypass* (%)” refere-se a taxa dos *bins* gerados por cada SE por meio do módulo *Bypass* em relação ao total de *bins* gerados por todos os SEs em todos os módulos de codificação aritmética. A coluna com o título ”Consec (%)” representa a porcentagem dos *bins* gerados pelo módulo *Bypass* processados consecutivamente. A coluna “BINS Consec (%)” está relacionada a porcentagem de *bins* que podem ser produzidos consecutivamente por meio do módulo *Bypass* relacionado ao total de *bins*. Na última coluna está representado a porcentagem de *bins* que podem ser gerados consecutivamente, tendo mais que ocorrências subsequentes (DEPRA, 2009).

5.4 Resultados e Análise da Codificação Aritmética sobre o H.264

A Tabela 13 apresenta a relação entre os elementos sintáticos codificados e o seu impacto na codificação.

Tabela 13 – Relação entre SEs e qualidade.

Tipo	VGA			HD720		
	Bitrate (%)	PSNR (dB)	Tempo (%)	Bitrate (%)	PSNR (dB)	Tempo (%)
COEF_LUM	+1,14	-0,08	-12,90	+2,40	-0,10	-18,79
COEF_CHR	+0,49	-0,05	-13,18	+1,20	-0,06	-19,24
QP_DELTA	+1,10	-0,05	-12,44	+1,94	-0,06	-18,02
INTRA_MODE	+1,04	-0,04	-12,64	+1,53	-0,05	-18,33
MVD_L0	+1,24	-0,07	-13,40	+2,90	-0,10	-19,51
MÉDIA	+1,00	-0,06	-12,91	+2,00	-0,07	-18,78

Os resultados demonstram que utilizando a relação dos elementos sintáticos agrupados o *bitstream* gerado crescerá em média 1% enquanto apresentará degradação visual de aproximadamente 0,06dB em uma codificação quase 13% mais rápida para os vídeos em resolução VGA. Considerando a resolução HD720 os resultados também apresentaram-se semelhantes, com *bitstream* aproximadamente 2% maior, degradação visual de 0,07dB e codificação 18% mais veloz.

5.5 Resultados e Análise da Codificação Aritmética sobre o HEVC

Existem diferenças na agrupação dos elementos sintáticos providos pelo H.264 e os providos pelo HEVC. A Tabela 14 mostra a relação de elementos sintáticos do HEVC de acordo com a sua partição.

Tabela 14 – Análise de elementos sintáticos no padrão HEVC.

Partição	Elementos Sintáticos
Control CC	end_of_slice_segment_flag, end_of_subset_one_bit, sao_merge_left_flag, sao_merge_up_flag, sao_type_idx_luma, sao_type_idx_chroma, split_cu_flag, cu_transquant_bypass_flag, cu_skip_flag, pred_mode_flag, part_mode, pcm_flag, prev_intra_luma_pred_flag, intra_chroma_pred_mode, rqt_root_cbf, merge_flag, merge_idx, inter_pred_idc, ref_idx_10, mvp_10_flag, ref_idx_11, mvp_11_flag, split_transform_ flag, cbf_luma, cbf_cb, cbf_cr, abs_mvd_greater0_flag, abs_mvd_greater1_flag, cu_qp_delta_abs, cu_chroma_qp_offset_flag, cu_chroma_qp_ offset_idx, log2_res_scale_abs_plus1, res_scale_sign_flag, transform_skip_flag, explicit_rdpcm_flag, explicit_rdpcm_dir_flag, last_sig_coeff_x_prefix, last_sig_coeff_y_prefix, coded_sub_block_flag
Control BC	sao_type_idx_luma, sao_type_idx_chroma, sao_offset_abs, sao_offset_sign, sao_band_position, sao_eo_class_luma, sao_eo_class_chroma, part_mode, mpm_idx, rem_intra_luma_pred_mode, intra_chroma_pred_mode, merge_idx, ref_idx_10, ref_idx_11, abs_mvd_minus2, mvd_sign_flag, cu_qp_delta_abs, cu_qp_delta_sign_flag, last_sig_coeff_x_suffix, last_sig_coeff_y_suffix
Luma Sig Map	Sig_coeff_flag
Luma Coeff Level	coeff_abs_level_greater1_flag, coeff_abs_level_greater2_flag
Luma BC	coeff_sign_flag, coeff_abs_level_remaining
Chroma Sig Map	sig_coeff_flag
Chroma Coeff Level	coeff_abs_level_greater1_flag, coeff_abs_level_greater2_flag
Chroma BC	coeff_sign_flag, coeff_abs_level_remaining

Todos os elementos sintáticos analisados geraram estatísticas para avaliação do comportamento dinâmico. Assim sendo, foi possível agregar os elementos que geram símbolos idênticos na construção do *bitstream*. Abaixo, a Tabela 15 e a Tabela 16 apresentam todos os elementos sintáticos agrupados e o impacto dos mesmos no processo de codificação aritmética.

Tabela 15 – Amostra de elementos sintáticos agrupados.

Tipo	VGA			HD720		
	Bitrate (%)	PSNR (dB)	Tempo (%)	Bitrate (%)	PSNR (dB)	Tempo (%)
Control CC	+0,49	-0,24	-32,27	+1,61	-0,4	-14,10
Control BC	+0,46	-0,69	-26,42	+1,13	-0,74	-31,06
Luma SigMap	+0,44	-0,66	-32,66	+2,34	-0,19	-29,13
Luma Coeff Level	+0,42	-0,96	-17,83	+1,25	-0,73	-18,15
Luma BC	+0,57	-0,88	-12,71	+1,95	-0,43	-36,54
Chroma SigMap	+0,45	-0,49	-15,30	+2,56	-0,89	-39,06
Chroma Coeff Level	+0,41	-0,27	-32,27	+1,61	-0,53	-14,10

Tabela 16 – Amostra de elementos sintáticos agrupados.

Tipo	HD1080			Ultra HD		
	Bitrate (%)	PSNR (dB)	Tempo (%)	Bitrate (%)	PSNR (dB)	Tempo (%)
Control CC	+1,16	-0,16	-39,00	+2,73	-0,45	-14,11
Control BC	+1,23	-0,28	-19,86	+2,03	-0,7	-28,92
Luma SigMap	+1,29	-0,18	-32,87	+1,18	-0,99	-14,51
Luma Coeff Level	+1,03	-0,6	-24,90	+2,94	-0,43	-15,85
Luma BC	+2,40	-0,02	-39,73	+2,75	-0,45	-16,48
Chroma SigMap	+3,00	-0,98	-27,39	+2,90	-0,99	-16,35
Chroma Coeff Level	+1,16	-0,25	-39,00	+2,73	-0,29	-14,11

A estratégia de utilização do agrupamento de elementos sintáticos na composição do *binstring* mostrou-se bastante efetiva. A Tabela 17 apresenta os resultados para percentual de erro comparando com a versão original. O máximo erro apresentado para composição de elementos agrupados foi de 9%. É importante verificar que o cenário de amostras não possui vídeos com corte de cena, desta forma o processo de agrupamento não é dinâmico na estratégia apresentada, sendo necessários ajustes no algoritmo para delimitar a sua atuação.

Tabela 17 – Percentual de acerto para amostras por rodadas de codificação.

Rodada	VGA	XGA	HD720	HD1080	Ultra HD
	%	%	%	%	%
1	97	97	100	94	96
2	91	93	92	100	96
3	100	99	98	95	97
4	98	93	97	93	91
5	97	93	91	95	95
6	98	99	91	92	98

A utilização de técnicas de simplificação permite atenuar o gargalo do sistema de codificação. Enquanto estratégias de codificação e decodificação para o H.264 atingiram um percentual de paralelismo de aproximadamente 30% com máxima de 47% para a codificação aritmética apresentados por (ROSA, 2010). Infelizmente não foi possível realizar análise simulada sobre o sistema computacional para calcular o máximo *throughput* para uma arquitetura utilizando esta técnica para o padrão HEVC. Através de análise estatística, estima-se que o máximo paralelismo chegue a 87% com média de 68%. Estas análises são baseadas nos dados de acertos e erros na predição dos elementos sintáticos de maior ocorrência.

6 Conclusão

A compressão de vídeo é de extrema importância para que seja possível a transmissão, o processamento e o armazenamento de vídeos digitais. Estudos relacionados a compressão de vídeo auxiliam no desenvolvimento de novas técnicas de compressão e no aprimoramento dos codificadores já em uso, seja em processadores genéricos, processadores gráficos bem como arquiteturas específicas para multimídia, buscando maior eficiência e ou rapidez na compressão dos vídeos.

Este trabalho apresenta uma análise do comportamento do fluxo de bits e dos elementos sintáticos da codificação aritmética adaptativa ao contexto no codificador de vídeo. A análise foi feita a partir das metodologias apresentadas no Capítulo 4 tendo como referência os trabalhos de (DEPRA, 2009) e (ROSA, 2010).

Os resultados obtidos demonstraram com segurança que a adoção de estratégias de geração de *binstrings* compostos por elementos sintáticos agrupados pode facilitar a adoção de técnicas de paralelismo. Conforme esperado, como efeito colateral da utilização da técnica, os resultados apresentaram uma suave degradação visual, bem como um pequeno aumento no *bitstream*. Para trabalhos futuros, alguns pontos devem ser amplamente analisados até a construção de uma arquitetura específica para codificação aritmética que seja capaz de contribuir efetivamente com sistemas computacionais de codificação de vídeos reais. Entre os principais pontos, destaca-se, implementação dinâmica para uma ampla análise de simulações com vídeos de cenas com cortes.

Referências

AGOSTINI, L. V. Desenvolvimento de arquiteturas de alto desempenho dedicadas à compressão de vídeo segundo o padrão h. 264/avc. 2007. Citado 5 vezes nas páginas 11, 15, 18, 19 e 27.

BOSSEN, F.; FLYNN, D.; SUHRING, K. Hm software manual. **Document: JCTVC-M1010**, 2013. Citado na página 36.

BOSSEN, F. et al. Common test conditions and software reference configurations. **JCTVC-L1100**, v. 12, 2013. Citado na página 48.

CHIMURKAR, M. P. High-throughput entropy encoder architecture for h.264/avc. 2015. Citado 2 vezes nas páginas 27 e 28.

CORREA, G. **Computational Complexity Reduction and Scaling for High Efficiency Video Encoders**. Tese (Doutorado), 2015. Citado 3 vezes nas páginas 20, 21 e 22.

DEPRA, D. A. Algoritmos e desenvolvimento de arquitetura para codificação binária adaptativa ao contexto para o decodificador h. 264/avc. 2009. Citado 11 vezes nas páginas 24, 30, 34, 35, 36, 41, 45, 52, 53, 54 e 58.

FILHO, O. M.; NETO, H. V. **Processamento digital de imagens**. [S.l.]: Brasport, 1999. Citado na página 13.

GONZALEZ, R. C.; WOODS, R. C. **Processamento digital de imagens .** [S.l.]: Pearson Educación, 2009. Citado na página 13.

IIDA, I.; WIERZZBICKI, H. A. Ergonomia. **Projeto e produção. 2ª ed. São Paulo: Edgard Blücher**, 2005. Citado na página 13.

ITU-T. International telecommunication union. **ITU-T Recommendation H.264 (04/2017): Advanced video coding for generic audiovisual services**, 2007. Citado na página 11.

ITU-T. International telecommunication union. **ITU-T Recommendation H.265 (02/2018): High efficiency video coding**, 2018. Citado na página 11.

ITU-T ISO/IEC. **Advanced Video Coding (H.264/AVC)**. 2018. Access date: 25 jun. 2018. Disponível em: <https://h264.hhi.fraunhofer.de/>. Citado na página 36.

ITU-T ISO/IEC. **High Efficiency Video Coding (HEVC)**. 2018. Access date: 25 jun. 2018. Disponível em: <https://hevc.hhi.fraunhofer.de/>. Citado na página 36.

KIM, Y. et al. Design of high performance arithmetic encoder for cabac in h. 264/avc. In: WORLD SCIENTIFIC AND ENGINEERING ACADEMY AND SOCIETY. **WSEAS International Conference. Proceedings. Mathematics and Computers in Science and Engineering**. [S.l.], 2009. Citado na página 29.

KOUMARAS, H.; KOURTIS, M.-A.; MARTAKOS, D. Benchmarking the encoding efficiency of h. 265/hevc and h. 264/avc. In: IEEE. **Future Network & Mobile Summit (FutureNetw), 2012**. [S.l.], 2012. p. 1–7. Citado 2 vezes nas páginas 31 e 32.

LI, H. **Gitl HEVC Analyzer**. 2018. Data de acesso: 17 set. 2018. Disponível em: <https://github.com/lheric/GitlHEVCAnalyzer>. Citado na página 46.

MIANO, J. **Compressed image file formats: Jpeg, png, gif, xbm, bmp**. [S.l.]: Addison-Wesley Professional, 1999. Citado na página 14.

POLLACK, J. Displays of a different stripe. **IEEE Spectrum**, IEEE, v. 43, n. 8, p. 40–44, 2006. Citado na página 13.

POURAZAD, M. T. et al. Hevc: The new gold standard for video compression: How does hevc compare with h. 264/avc? **IEEE consumer electronics magazine**, IEEE, v. 1, n. 3, p. 36–46, 2012. Citado 3 vezes nas páginas 19, 21 e 22.

RICHARDSON, I. E. **The H. 264 advanced video compression standard**. [S.l.]: John Wiley & Sons, 2011. Citado 4 vezes nas páginas 14, 16, 25 e 28.

ROSA, V. S. d. Arquiteturas de hardware dedicadas para codificadores de vídeo h. 264: filtragem de efeitos de bloco e codificação aritmética binária adaptativa a contexto. 2010. Citado 10 vezes nas páginas 14, 36, 37, 39, 40, 41, 42, 43, 57 e 58.

SALOMON, D. **Data compression: the complete reference**. [S.l.]: Springer Science & Business Media, 2004. Citado na página 25.

SHI, Y. Q.; SUN, H. **Image and video compression for multimedia engineering: Fundamentals, algorithms, and standards**. [S.l.]: CRC press, 1999. Citado 2 vezes nas páginas 15 e 26.

SILVA, T. L. d. **Contribuições para a Redução de Complexidade da Codificação Intra na Norma HEVC e Extensão 3D-HEVC**. Tese (Doutorado), 2015. Citado 3 vezes nas páginas 20, 22 e 23.

SILVEIRA, B. S. d. C. d. et al. Exploração arquitetural nas métricas de similaridade para codificadores de vídeo do padrão hevc. Universidade Catolica de Pelotas, 2016. Citado 2 vezes nas páginas 20 e 22.

SULLIVAN, G. J. et al. Overview of the high efficiency video coding (hevc) standard. **IEEE Transactions on circuits and systems for video technology**, IEEE, v. 22, n. 12, p. 1649–1668, 2012. Citado na página 19.

SZE, V.; BUDAGAVI, M. High throughput cabac entropy coding in hevc. **IEEE Transactions on Circuits and Systems for Video Technology**, IEEE, v. 22, n. 12, p. 1778–1791, 2012. Citado 4 vezes nas páginas 23, 32, 33 e 34.

THIELE, C. Desenvolvimento da arquitetura dos codificadores de entropia adaptativos cavlc e cabac do padrão h. 264/avc. 2012. Citado 2 vezes nas páginas 17 e 28.

TOURAPIS, A. M. et al. H. 264/14496-10 avc reference software manual. **Doc. JVT-AE010**, 2009. Citado na página 36.

YAMADA, F. et al. Parte i-sistemas de tv digital. **Revista Mackenzie de Engenharia e Computação**, v. 5, n. 5, 2010. Citado 2 vezes nas páginas 15 e 16.