UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação

Tese

Controle Adaptativo de Complexidade do Codificador AV1 baseado em Frente de Pareto e Aprendizado de Máquina

Isis Duarte Bender

Pelotas, 2023

Isis Duarte Bender

Controle Adaptativo de Complexidade do Codificador AV1 baseado em Frente de Pareto e Aprendizado de Máquina

Tese apresentada ao Programa de Pós-Graduação em Computação do Centro de Desenvolvimento Tecnológico da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Doutor em Ciência da Computação.

Orientador: Prof. Dr. Marcelo Schiavon Porto
Coorientador: Prof. Dr. Guilherme Ribeiro Corrêa
Colaborador: Prof. Dr. Luciano Volcan Agostini

Pelotas, 2023

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

B458c Bender, Isis Duarte

Controle adaptativo de complexidade do codificador AV1 baseado em frente de pareto e aprendizado de máquina / Isis Duarte Bender ; Marcelo Schiavon Porto, orientador ; Guilherme Ribeiro Corrêa, Luciano Volcan Agostini, coorientadores. — Pelotas, 2023.

201 f. : il.

Tese (Doutorado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2023.

1. Codificação de vídeo. 2. AV1. 3. Complexidade. 4. Controle adaptativo. I. Porto, Marcelo Schiavon, orient. II. Corrêa, Guilherme Ribeiro, coorient. III. Agostini, Luciano Volcan, coorient. IV. Título.

CDD : 005

Elaborada por Maria Inez Figueiredo Figas Machado CRB: 10/1612

Isis Duarte Bender

Controle Adaptativo de Complexidade do Codificador AV1 baseado em Frente de Pareto e Aprendizado de Máquina

Tese aprovada, como requisito parcial, para obtenção do grau de Doutor em Ciência da Computação, Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas.

Data da Defesa: 23 de março de 2023

Banca Examinadora:

Prof. Dr. Marcelo Schiavon Porto (orientador)
Doutor em Ciência da Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Bruno Zatt
Doutor em Engenharia Elétrica pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Mateus Grellert da Silva
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Pedro Antônio Amado Assunção
Doutor em Sistemas de Computação pela Universidade de Essex - (Reino Unido).

Dedico esta conquista à pessoa que amo incondicionalmente: meu filho Diogo! Tu és a minha razão de procurar evoluir a cada dia.

AGRADECIMENTOS

Para desenvolver este tralho muitos desafios tiveram que ser superados. Se não bastasse a total inexperiência na área de codificação de vídeo, grande parte do trabalho precisou ser desenvolvido à distância devido a pandemia do COVID-19. Porém, ao longo do caminho diversas pessoas contribuíram para a realização deste sonho e de forma alguma, poderia deixar de agradecê-las.

Primeiramente, gostaria de agradecer ao meu orientador Prof. Dr. Marcelo Porto por ser bastante acessível e estar sempre disposto a me auxiliar. Obrigada pelos ensinamentos e confiança durante esse período. Agradeço também aos coorientadores Guilherme Corrêa e Luciano Agostini pela atenção e apoio dado. Aproveito para parabenizar todos vocês pelo excelente trabalho que realizam, no qual prevalece o espírito de equipe.

Agradeço aos colegas do grupo de pesquisa VITECH (*Video Technology Research Group*), sempre dispostos a colaborar. Com certeza é um grupo admirável! Um agradecimento especial ao Alex Borges e ao Gustavo Rehbein, pelas ajudas e contribuições diretas no trabalho. Também não poderia deixar de agradecer a todos os colegas e amigos da coordenadoria de Eletrônica, especialmente, a Roberta de Carvalho Nobre Palau.

Agradeço os meus pais e a minha irmã pelo incentivo e o apoio constantes em todas as etapas da minha vida. Amo vocês!

Por fim, agradeço a todos que de uma forma ou de outra torceram por mim.

Educar a mente sem educar o coração não é educação.
— ARISTÓTELES

RESUMO

BENDER, Isis Duarte. **Controle Adaptativo de Complexidade do Codificador AV1 baseado em Frente de Pareto e Aprendizado de Máquina**. Orientador: Marcelo Schiavon Porto. 2023. 203 f. Tese (Doutorado em Ciência da Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2023.

Os codificadores, responsáveis pela compressão dos vídeos digitais, viabilizam a utilização deste tipo de mídia, pois reduzem a taxa de dados, mantendo a qualidade da imagem. O contínuo desenvolvimento e as melhorias dos padrões de codificação de vídeos digitais são essenciais para impulsionar aplicativos baseados neste tipo de mídia, fazendo com que o mercado, constantemente, procure novos algoritmos e padrões de codificação que alcancem alta eficiência de compressão. Diante deste cenário, a AOM (*Alliance for Open Media*) desenvolveu o AV1 (AOMedia Video 1), codificador de vídeo com uma alta taxa de compressão. No entanto, as ferramentas avançadas e os aprimoramentos presentes no AV1 apresentam um alto custo computacional, acarretando um elevado tempo de codificação. Diante disso, o principal objetivo deste trabalho é contribuir com estratégias para o controle adaptativo de complexidade do codificador AV1, visando otimizar a relação entre tempo e eficiência de codificação para vídeos de resolução HD 1080 e UHD 4K. Para isso, um estudo sobre a complexidade do codificação do AV1 foi realizado, onde diferentes os pontos de controle foram obtidos a partir da análise de parâmetros de codificação do AV1. A partir disso, a Versão *Baseline* do controlador foi desenvolvida e usada para avaliar a viabilidade do desenvolvimento do controlador adaptativo de complexidade AV1. Visando o aperfeiçoamento da capacidade de controle dinâmico do controlador, uma nova versão do controlador foi desenvolvida, chamada de *CCAM*. Nela, modelos de aprendizado de máquina foram utilizados para a predição do tempo de codificação e classificação do vídeo de entrada conforme suas características. Os resultados de erro médio obtidos com controlador *CCAM* variam de 0,11 a 1,88 pontos percentuais na resolução HD 1080 e de 0,14 a 3,33 pontos percentuais na resolução UHD 4K, para uma faixa de redução de tempo de codificação de 10% a 70%. Já os valores de BD-Rate médio, obtidos para controlador *CCAM* nas resoluções HD 1080 e UHD 4K, encontram-se entre 2,63% e 54,57% e 2,08% e 38,30%, respectivamente. Além disso, resultados de uma avaliação subjetiva de qualidade apontam uma mínima degradação na qualidade subjetiva das sequências comprimidas a partir do controlador *CCAM* desenvolvido. Cabe destacar ainda que os controladores propostos nesta tese são as primeiras soluções existentes na literatura para o controle adaptativo de complexidade do codificador AV1.

Palavras-chave: Codificação de vídeo. AV1. Complexidade. Controle Adaptativo.

ABSTRACT

BENDER, Isis Duarte. **Titulo do Trabalho em Ingles**. Advisor: Marcelo Schiavon Porto. 2023. 203 f. Thesis (Doctorate in Computer Science) – Technology Development Center, Federal University of Pelotas, Pelotas, 2023.

The encoders, responsible for the compression of the digital videos, enable the use of this type of media, as they reduce the data rate, maintaining the image quality. The continuous development and Improvements to digital video coding standards are essential to boost applications based on this type of media, making the market, constantly look for new algorithms and coding standards that reach high compression efficiency. Given this scenario, the AOM (Alliance for Open Media) developed AV1 (AOMedia Video 1), video encoder with a high compression ratio. However, advanced tools and enhancements present in AV1 have a high computational cost, leading to a high coding time. Therefore, the main objective of this work is to contribute with strategies for adaptive complexity control of the AV1 encoder, aiming to optimize the relationship between time and coding efficiency for HD 1080 and UHD 4K resolution videos. For this, a study on the complexity of AV1 coding was carried out, where different points of control were obtained from the analysis of AV1 coding parameters. From this, the *Baseline* Version of the controller was developed and used to evaluate the viability of developing the adaptive AV1 complexity controller. To improve the dynamic control capacity of the controller, a new version of the controller was developed, called *CCAM*. In it, machine learning models were used to predict the coding time and classify the input video according to its characteristics. The average error results obtained with the *CCAM* controller range from 0.11 to 1.88 percentage points at HD 1080 resolution and from 0.14 to 3.33 percentage points at UHD 4K resolution, for a range of encoding time reduction from 10% to 70%. The average BD-Rate values obtained for the *CCAM* controller in HD 1080 and UHD 4K resolutions are between 2.63% and 54.57% and 2.08% and 38 .30%, respectively. Furthermore, results of a subjective quality assessment indicate a minimal degradation in the subjective quality of the compressed sequences from the *CCAM* controller developed. It should also be noted that the controllers proposed in this thesis are the first solutions in the literature for the adaptive control of the complexity of the AV1 encoder.

Keywords: Video coding. AV1. Complexity. Adaptative Control.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADST	Asymmetrical Discrete Sine Transform
AOM	Alliance for Open Media
APOC	Atualização de Pontos de Operação do Codificador
AV1	AOMedia Video 1
AVC	Advanced Video Coding
BD	Bjøntegaard Difference
BD-PSNR	Peak Signal-to-Noise Ratio
BD-Rate	Bjøntegaard Rate Difference
Cb	Crominância azul
CCAM	Controlador de Complexidade baseado em Aprendizado de Máquina
CDEF	Constrained Directional Enhancement Filter
CfL	Chroma from Luma
CNN	Convolutional Neural Network
COVID	Corona Virus Disease
CP	Conjunto de parâmetro
CQ	Constant Quality
Cr	Crominância vermelha
CRFPM	Compound Reference Frame Prediction Mode
CTC	Common Test Coditions
CTU	Coding Tree Unit
CU	Coding Unit
DBF	Deblocking Filter
DCR	Degradation Category Rating
DCT	Discrete Cosine Transform
DP	Desvio Padrão
FRC	Fator de Redução de Complexidade

GFG	Golden Frame Group
HD	High Definition
HEVC	High Efficiency Video Coding
IDTX	Identity Transform
IntraBC	Intrablock Copy
LCU	Largest Coding Unit
Libaom	AV1 Codec Library
MC	Motion Compensation
ME	Motion Estimation
MIMO	Multiple Input Multiple Output
MOS	Mean Opinion Score
MSE	Mean Squared Error
N_Q	Número de quadros da sequência de vídeo
OBMC	Overlapped Block Motion Compensation
OE	Objetivos Específicos
PC	Ponto de Controle
PID	Proporcional Integral Diferencial
PSNR	Peak Signal to Noise Ration
QTMT	Quad-Tree plus Multi-Type Tree
RA	Random Acess
R-D	Rate-distortion
RDM	Rate Distortion Model
RDO	Rate Distortion Optimization
RGB	Red Green Blue
R_T	Relação de tempo
R_{TA}	Relação de tempo ajustada
RTC	Redução de Tempo de Codificação
SAD	Sum of Absolute Differences
SATD	Sum of Absolute Transformed Differences
SB	Super Block
SI	Spatial Index
SISO	Single Input Single Output
SLRF	Switchable Loop Restoration Filter
SOS	Standard Deviation of Score

SP	Set Point
SRFPM	Single Reference Frame Prediction Mode
SSE	Sum Square Error
SW	Software
T_{CP}	Tempo com a configuração padrão
T_D	Tempo de codificação desejado
T_{DA}	Tempo de codificação desejado ajustado
T_{GO}	Tempo de codificação do grupo observado
T_{GP}	Tempo de codificação do grupo predito
T_{GPA}	Tempo de codificação do grupo predito ajustado
T_Q	Tempo de codificação do quadro
T_{QP}	Tempo de codificação do quadro predito
TI	Temporal Index
UHD	Ultra-High Definition
VITECH	Video Technology Research Group
VVC	Versatile Video Coding
Y	Luminância

SUMÁRIO

1 INTRODUÇÃO

Ao longo das últimas décadas, a indústria de semicondutores, acompanhada pela indústria eletrônica, vem evoluindo de forma significativa. Atualmente, características como versatilidade, complexidade, confiabilidade e miniaturização são encontradas nos produtos e, frequentemente, são desenvolvidos equipamentos que agregam novas funcionalidades. O telefone celular, por exemplo, além de ter suas dimensões reduzidas, passou a oferecer aos usuários vários recursos, tais como: câmera fotográfica/de vídeo, geolocalização, leitor de impressão digital, acesso à internet, entre outros. Além das telecomunicações, a eletrônica embarcada se faz presente em outras áreas, como o entretenimento e bens de consumo, podendo ser encontrada facilmente no dia a dia da maioria das pessoas.

Este contexto, aliado também à popularização das redes sociais, contribuiu para que vídeos digitais se tornassem cada vez mais populares. Atualmente, vídeos digitais de alta definição (*HD – High Definition*) e ultra-alta definição (*UHD – Ultra-High Definition*) encontram-se presentes no cotidiano das pessoas, sendo acessados e reproduzidos, frequentemente, em dispositivos móveis como smartphones, tablets e notebooks. Uma das plataformas de vídeo online mais conhecidas da atualidade, o YouTube, possui mais de dois bilhões de usuários ativos por mês (YOUTUBE, 2022). Outro fator de grande impacto no crescimento do consumo de *streaming* de vídeo foi a pandemia do COVID-19. Em um curto período, de fevereiro a março de 2020, a empresa Bitmovin constatou que o número de vídeos reproduzidos aumentou 118%, e o tempo de vídeos assistidos na internet aumentou 200% (BITMOVIN, 2020). Um estudo mostra que o crescimento médio da transmissão ao vivo do Facebook foi de até 200% na Europa e 300% nos EUA (BÖTTGER; IBRAHIM; VALLIS, 2020). A representatividade do mercado global de *streaming* é bastante expressiva. Em 2021, este mercado foi avaliado em 59,14 bilhões de dólares e deve se expandir a uma taxa de crescimento anual composta, métrica que mede a taxa de retorno de um determinado investimento, de 21,3% de 2022 até 2030 (RESEACH, 2022).

Para apresentar maior qualidade, tornando as imagens mais próximas da realidade, os vídeos digitais utilizam uma quantidade bastante significativa de dados (bits),

o que traz consequências para o armazenamento, para a transmissão e para o esforço computacional utilizado na reprodução. Dessa forma, a compressão dos dados necessários para a representação de vídeo é indispensável para viabilizar a reprodução, armazenamento e transmissão deste tipo de mídia, mantendo a qualidade e utilizando taxas muito menores de dados. A compressão dos dados é realizada pelos codificadores através da exploração das redundâncias espacial e temporal presentes nos vídeos digitais.

Ao longo do tempo, a crescente necessidade de promover o intercâmbio de vídeos digitais entre os diversos dispositivos eletrônicos existentes no mercado, fez com que padrões de compressão ganhassem a atenção tanto da indústria como de pesquisadores do meio acadêmico. O contínuo desenvolvimento e as melhorias dos padrões de codificação de vídeo são essenciais para impulsionar aplicativos baseados neste tipo de mídia, fazendo com que o mercado constantemente procure novos algoritmos e codificadores que alcancem maior eficiência de compressão.

Muitos padrões de codificação de vídeo foram desenvolvidos nas últimas décadas. Os reconhecidos grupos de padronização ITU-T e ISO/IEC são responsáveis pelo desenvolvimento de importantes codificadores que utilizam tecnologias patenteadas, estando vinculados a uma taxa de licenciamento para uso, entre eles: o H.264/AVC (*Advanced Video Coding*) (ISO/IEC, 2003) (ITU-T, 2003), o H.265/HEVC (*High Efficiency Video Coding*) (ISO/IEC, 2013) (ITU-T, 2013) e H.266/VVC (*Versatile Video Coding*) (BROSS et al., 2020). No entanto, os vídeos digitais são frequentemente acessados por aplicativos que estão sendo executados na internet. Como as principais tecnologias da internet são abertas, a Google embarcou no projeto WebM (MUKHERJEE et al., 2013), o qual tinha como objetivo desenvolver codecs com código aberto e livre de royalties (WEBM, 2012). O codec pioneiro deste projeto foi lançado em 2010 e intitulado VP8, o qual foi sucedido pelo VP9 (PARKER et al., 2016) em 2013. Em 2015, a Google uniu esforços com diversas empresas de alta tecnologia, entre elas, a Cisco e a Xiph, empresas responsáveis, respectivamente, pelo desenvolvimento dos codificadores Thor e Daala, formando uma aliança conhecida como AOMedia (*Alliance for Open Media*) (AOMEDIA, 2019). Atualmente, cerca de trinta empresas de alta tecnologia compõem esta aliança, dentre elas: AMD, ARM, Intel, Nvidia, Microsoft, Mozilla, Adobe, Amazon, Netflix, entre outras (AOMEDIA, 2022). A AOM é a responsável pela criação do novo codec chamado AV1 (RIVAZ; HAUGHTON, 2018), que foi lançado em 2018.

Inquestionavelmente, os avanços ocorridos nos padrões de codificação no decorrer dos anos geraram melhorias na eficiência do processo de codificação de vídeos, ou seja, podem resultar em vídeos codificados com um menor volume de bits, ao mesmo tempo em que mantêm a qualidade visual mais próxima possível ao vídeo original. Em contrapartida, para atingir tais avanços, proporcionando aos usuários vídeos

digitais com alta qualidade de imagem e elevada taxa de compressão, os codecs dispõem de diversas ferramentas com alto grau de complexidade ao longo do processo de codificação. Neste trabalho, complexidade será tratada como sinônimo a custo computacional, como ocorre nos demais trabalhos da área de codificação de vídeos. Pesquisas a respeito das possibilidades relacionadas à redução de complexidade nos codificadores, como a desenvolvida por (CORRÊA et al., 2011), são extremamente pertinentes, pois o tempo total de processamento, na grande maioria dos casos, está diretamente relacionado com o custo computacional existente no processo de codificação. A elevada complexidade dos codificadores implica no incremento do consumo de energia do dispositivo, fato extremamente indesejável, principalmente para dispositivos portáteis alimentados por bateria.

O AV1, assim como os demais codificadores atuais, disponibiliza um número elevado de opções de processamento que devem ser avaliadas durante o tempo de codificação para a definição da opção que confere o melhor resultado em termos de eficiência de compressão. Projetado com o objetivo de alcançar taxas de compressão mais altas em comparação aos codificadores anteriores, o desenvolvimento do codificador AV1 teve como ponto de partida o VP9 (NGUYEN; MARPE, 2018). Ao longo do tempo, novas ferramentas foram introduzidas e outras otimizadas, tornando o processo de codificação do AV1 ainda mais complexo. O AV1 oferece uma redução de quase 30% no BD-Rate (*Bjøntegaard Rate Difference*) (BJØNTEGAARD, 2001) em relação ao seu predecessor, o VP9 (CHIANG; HAN; XU, 2019). No entanto, para a obtenção deste ganho, o AV1 introduziu/expandiu diversas ferramentas de compressão.

1.1 Motivação

O incremento significativo das ferramentas de codificação do AV1, contribuem para a elevada complexidade envolvida no seu processo de codificação. Esta complexidade pode ser vista como uma desvantagem para a estabilização do novo codificador no mercado, pois influencia diretamente no tempo de codificação. Tanto o artigo da autora desta tese (BENDER et al., 2019) quanto (MANSRI et al., 2020) comparam o tempo de codificação do software de referência do AV1, Libaom (*AV1 Codec Library*), em relação a outros codificadores existentes no mercado e apontam que o Libaom requer o maior tempo de codificação em todos os casos analisados. Segundo Mansri et al. (2020) o Libaom só é mais rápido que o VTM (software de referência do VVC), que foi lançado dois anos depois, em 2020. A comparação do tempo de execução de vários codificadores, realizada em Mansri et al. (2020) é apresentada na Tabela 1. O tempo de execução de codificadores específicos aos formatos/padrões VVC, AV1 e VP9 são normalizados em relação ao tempo de execução do padrão HEVC. O AV1 é

2,25 vezes mais rápido que o VVC e, aproximadamente, 27 vezes mais lento que o codificador VP9. Em uma avaliação desenvolvida no escopo deste trabalho e publicada em (BENDER et al., 2019), pôde ser observado um elevado tempo de codificação requerido pelo codificador AV1 em relação ao HEVC. Para os experimentos realizados, os resultados mostram que o Libaom, software de referência do AV1 (AOMEDIA, 2018a), apresenta um custo computacional 14,64 vezes maior e um acréscimo no BD-Rate de 16,35% em comparação com o HM, software de referência do HEVC.

Tabela 1 – Tempo normalizado de codificação, para configuração *low-delay*, tendo como âncora o padrão HEVC. Adaptado de (MANSRI et al., 2020).

Codificador	**SW de Referência**	**Tempo Normalizado de Codificação**
VVC	VTM v8.0	15,30
AV1	Libaom v1.0.0	6,80
VP9	v1.8.2-98-gc2aa152	0,25

Por outro lado, o novo codificador apresenta características que geram grandes expectativas referentes ao seu sucesso, como por exemplo, o ganho de compressão e o apoio de importantes empresas de tecnologia. A Intel lançou um codec que suporta o padrão AV1, denominado SVT-AV1 (ARMASU, 2019). A Netflix está usando o AV1 no Android (GUO et al., 2020). A NETINT anunciou o primeiro hardware disponível comercialmente para o codificador AV1 voltado a data center (NETINT, 2021). Já a NVIDIA possui unidade de processamento gráfico que fornece decodificação para o AV1 (NVIDIA, 2021). Assim, diante do contexto apresentado e, aliado ao recente lançamento do AV1, é possível concluir que estudos que gerem contribuições referentes à redução de complexidade no AV1 possuem grande relevância no cenário atual e também no cenário futuro da área, uma vez que a família de codificadores que a AOM pretende gerar deverá estar fortemente presente no mercado, considerando que as principais empresas da área, desde empresas de hardware até empresas de streaming, fazem parte da AOM e pretendem usar o AV1 em seus produtos.

Nesse cenário, pesquisas voltadas ao controle dinâmico de complexidade podem ser consideradas ainda mais promissoras, pois além de promover a redução de complexidade, pode fazer isso de maneira controlada. Ou seja, permite que a complexidade seja reduzida, em tempo de execução, a partir das características do vídeo ou de configurações de usuário, por exemplo. Isso pode proporcionar maior eficiência na relação redução da complexidade versus qualidade do vídeo codificado, bem como permitir a variação do alvo de redução de complexidade durante a codificação. Considerando um dispositivo portátil alimentado por bateria, os recursos computacionais requeridos para a codificação podem ser ajustados de acordo com o nível da bateria do dispositivo. Dessa forma, se o nível da energia da bateria estiver baixo, a complexidade de codificação pode ser reduzida, diminuindo, consequentemente, o tempo de execução e o consumo de energia, no entanto, com inevitável perda de eficiência de

codificação. Num cenário oposto, caso não haja restrições de energia, os recursos computacionais usados na codificação podem ser expandidos, proporcionado maior eficiência de codificação, mas acarretando num maior tempo de processamento e consumo de energia.

A literatura disponibiliza uma variedade considerável de trabalhos que propõem estratégias para controlar a complexidade da codificação de vídeo. Grande parte desses trabalhos está relacionado ao padrão de vídeo HEVC, como (CORRÊA et al., 2011), (JIMENEZ-MORENO; MARTÍNEZ-ENRÍQUEZ; MARÍA, 2016) e (DENG et al., 2016). No entanto, cabe destacar que, até o presente momento, não há na literatura solução específica para o controle de complexidade do codificador AV1. Considerando a discussão apresentada, este trabalho está focado na seguinte questão de pesquisa:
É possível controlar, de maneira eficiente, a complexidade do codificador AV1? Nesse caso, a eficiência do controlador poderá ser avaliada considerando os níveis de redução de complexidade dos seus pontos de operação e as perdas de eficiência de codificação que cada ponto de operação apresenta?
Para tentar responder esta questão, a principal hipótese investigada foi:
Visto que os parâmetros utilizados na codificação impactam diretamente no tempo de codificação, a seleção correta, e dinâmica, destes parâmetros pode levar a um controle eficiente da complexidade do AV1, desde que seja aplicado um método de seleção que mantenha uma relação satisfatória entre RTC (Redução de Tempo de Codificação) e as perdas de eficiência de codificação.
Esta hipótese foi investigada no decorrer do trabalho, o qual enfrentou diversos desafios devido à falta de documentação relacionados ao codificar AV1, principalmente, no período inicial do desenvolvimento deste trabalho. Além disso, como poderá ser observado no Capítulo 4, não existem trabalhos publicados na literatura propondo soluções para o controle de complexidade do AV1, impossibilitando uma comparação justa dos resultados obtidos nesta tese com outro sistema de controle.

1.2 Objetivos e Contribuições

Diante da hipótese apresentada na seção anterior, o principal objetivo desta tese é o desenvolvimento de um controlador adaptativo de complexidade para o codificador AV1 que, em tempo de execução, atue no processo de codificação de modo a efetuar o controle de complexidade do codificador, selecionando pontos de operação que otimizem a relação entre redução de complexidade e perdas na eficiência de codificação. Para que o objetivo geral seja realizado, três objetivos específicos (OE) foram traçados:

- **OE1** - Compreensão do impacto de complexidade promovido por cada etapa de codificação AV1 para direcionar os esforços de otimização aplicados durante o

controle adaptativo de complexidade.

- **OE2** – Desenvolvimento de controle adaptativo de complexidade do AV1 com base em um conjunto de parâmetros extraídos do codificador, a partir da adaptação de um modelo de controlador de complexidade existente na literatura.
- **OE3** – Desenvolvimento de controle adaptativo de complexidade do AV1 que tome decisões com base em predição temporal e explore diferentes características dos vídeos a partir do uso de algoritmos de aprendizado de máquina.

Como principal contribuição desta tese, deve ser destacado o desenvolvimento da primeira solução de controle dinâmico de complexidade para o codificador AV1.

1.3 Apresentação do Texto

Esta tese está organizada em oito capítulos. O Capítulo 2 apresenta o princípio de codificação de vídeos digitais. O Capítulo 3 trata do codificador AV1. O Capítulo 4 refere-se aos trabalhos relacionados ao tema abordado nesta proposta. O Capítulo 5 apresenta resultados referentes à investigação inicial sobre o comportamento do AV1. O Capítulo 6 apresenta o controlador de complexidade adaptativo desenvolvido para o formato AV1. O Capítulo 7 apresenta os resultados do sistema de controle adaptativo de complexidade para AV1. Por fim, o Capítulo 8 apresenta as conclusões.

2 CONCEITOS BÁSICOS SOBRE CODIFICAÇÃO DE VÍDEOS DIGITAIS

Os vídeos digitais são formados por uma sequência de imagens (quadros) estáticas independentes e capturadas em uma determinada taxa de amostragem (quadros por segundo) para que promova a sensação de movimento ininterrupto ao espectador, como mostra a Figura 1(a). Para uma sensação de movimento contínuo, esta taxa de amostragem deve ser de pelo menos 24 a 30 quadros por segundo (GONZALEZ; WOODS, 2003). No entanto, para vídeos de alta e ultra alta definição, são comuns taxas de amostragem ainda maiores, como 60 ou 120 quadros por segundo. A resolução do vídeo é dada pela quantidade de pixel nas dimensões horizontal (x) e vertical (y) de uma imagem. Quanto maior a quantidade de pixel, maior a sua resolução e, consequentemente, maior será a quantidade de bits necessários para enviá-la ou armazená-la.

Para o processo de codificação, os quadros são divididos em blocos. Os blocos podem assumir tamanhos distintos e serem constituídos por diferentes quantidades de pixels, como pode ser observado na Figura 1(b). O pixel é considerado a menor informação que compõe uma imagem digital e pode conter até três informações de cores. Cada uma dessas informações, denominada amostra, pode ser tratada, separadamente, pelos codificadores para atingir taxa de compressão mais altas.

2.1 Redundâncias na Representação de Vídeos

Apesar dos vídeos digitais necessitarem de uma quantidade elevada de dados para serem representados, esse tipo de mídia possui um alto grau de redundância. A redundância presente nos vídeos digitais pode ser classificada como: redundância espacial, redundância temporal e redundância estatística (AKRAMULLAH, 2014). De maneira sucinta, pode-se dizer que a primeira corresponde à correlação existente entre os pixels espacialmente distribuídos em um mesmo quadro, isto pode ser observado na Figura 1(b). No bloco 4x4, por exemplo, todos os pixels estão altamente relacionados, ou seja, são bastante semelhantes. A segunda é causada pela corre-

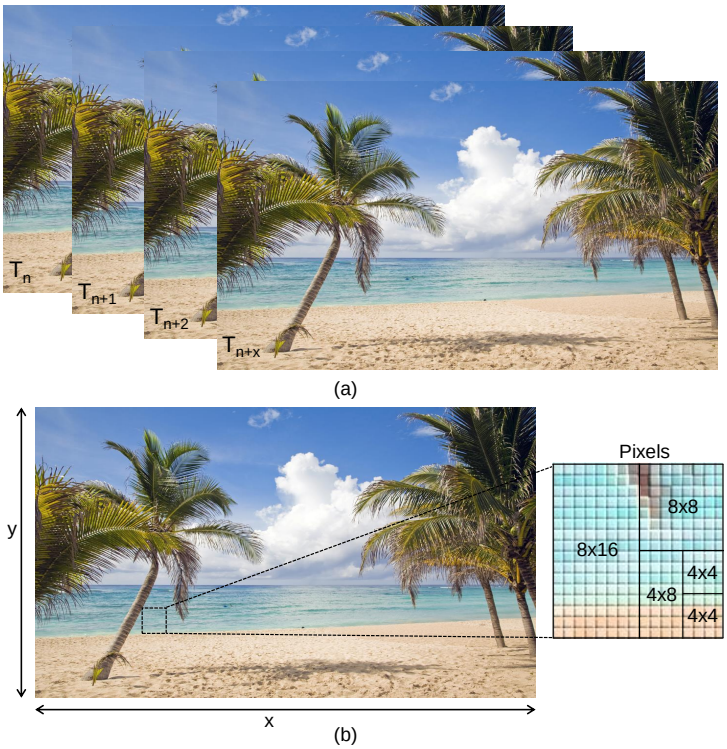


Figura 1 – Conceitos de codificação de vídeo. (a) Redundância Temporal. (b) Exemplo de particionamento de um quadro.

lação existente entre quadros temporalmente próximos, como pode ser observado na Figura 1(a). Por fim, a terceira está relacionada com a probabilidade de ocorrência dos símbolos codificados.

A eliminação das informações redundantes é uma das principais técnicas exploradas nos padrões atuais de codificação para obter ganhos de compressão, sem impactar na percepção visual. Para imagens ricas em detalhes, cerca de 46% das informações são redundantes e este valor é ainda mais expressivo, cerca de 74%, quando se trata de imagens com poucos detalhes (AKRAMULLAH, 2014).

2.2 Espaço de Cores e Subamostragem

Além disso, a compressão pode ser ainda mais otimizada fazendo uso de subamostragem no devido espaço de cores. Segundo Richardson (2002), os dois espaços de cores mais comuns são: o RGB (***R***ed, ***G***reen, ***B***lue) e o YCbCr. No espaço de cores RGB as amostras são representadas a partir das três cores primárias (vermelho, verde e azul). As três componentes de cores são igualmente importantes e vinculadas à luminosidade. Já no espaço de cores YCbCr, a informação de luminância (Y) é independente das informações de crominância, azul e vermelha (respectivamente, Cb e Cr).

O sistema visual humano é mais sensível à informação de luminosidade (brilho) do que à informação de cores (RICHARDSON, 2002), logo, a possibilidade de tratar

individualmente os componentes de luminância (luma) e crominância (croma) torna o espaço de cores YCbCr mais indicado para o processo de codificação. Outra vantagem é a possibilidade de representar os componentes Cr e Cb com uma resolução menor que o componente Y, sem gerar um impacto expressivo na qualidade visual.

Como pode ser observado na Figura 2, o espaço de cores YCbCr pode apresentar diferentes formatos, como: 4:4:4, 4:2:2 e 4:2:0 (RICHARDSON, 2002). No primeiro caso, tanto as informações de luminância quanto as de crominância são representadas com a mesma proporção, ou seja, não há descarte de informação. Porém, nos dois casos seguintes, a informação de crominância é subamostrada. No formato 4:2:2, para cada quatro informações de luma, há duas informações Cb e duas informações Cr. Já no formato 4:2:0, para cada quatro informações de luminância, há uma informação relacionada à Cb e outra à Cr. A subamostragem das informações de crominância pode ser considerada uma das primeiras etapas do processo de compressão de vídeo a partir da eliminação de informações menos relevantes ao sistema visual humano. No formato 4:2:2, um terço das informações de crominância são descartadas, antes mesmo de qualquer etapa do processo de compressão de vídeo. Já no formato 4:2:0, metade destas informações são desprezadas.

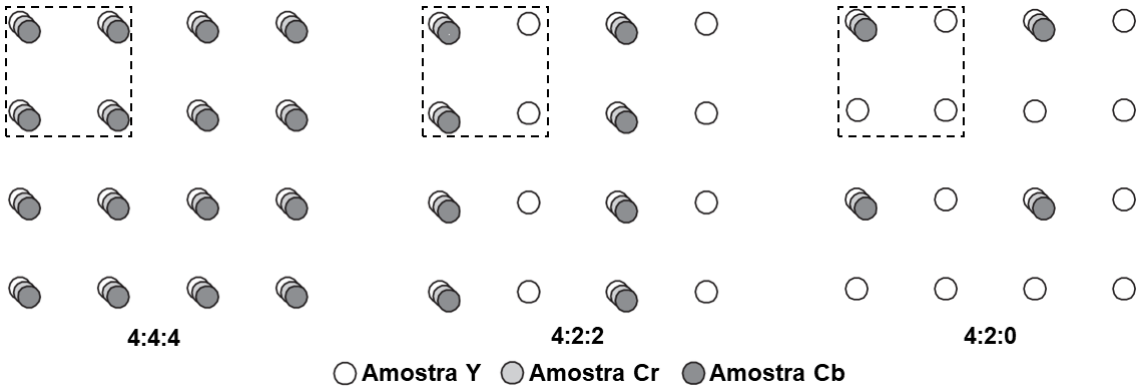


Figura 2 – Formatos do espaço de cores YCbCr. Adaptado de (RICHARDSON, 2002).

2.3 Otimização Taxa-Distorção

Durante o processo de codificação, diversas decisões devem ser tomadas pelo codificador, como a escolha do tamanho do bloco usado nas etapas de predição e transformadas, o tipo de predição (intra-quadro ou inter-quadros), os modos de predição, etc. Esses tipos de decisões são realizados frequentemente pelo codificador, visto que os vídeos a serem codificados apresentam variação de conteúdo e movimento. Para otimizar a eficiência de codificação, o codificador deve selecionar, entre as inúmeras possibilidades de codificação, as que melhor se adéquam para a região do vídeo a ser codificado.

A decisão do melhor modo de codificação é baseada na técnica conhecida como

Rate Distortion Optimization (RDO), a qual é definida pela equação 1. Os algoritmos RDO selecionam a opção com melhor relação entre taxa e distorção, ou seja, a que gera o menor custo (J) entre ambas as métricas. A relação entre taxa (R) e distorção (D) é controlada pelo multiplicador de Lagrange λ o qual é definido a partir de aproximações empíricas (RICHARDSON, 2002).

$$J = D + \lambda.R \tag{1}$$

2.4 Métricas de Comparação

A existência de critérios que permitam a comparação entre o vídeo comprimido e o vídeo original é importante para avaliar a eficiência de codificação, a qual está relacionada não só à capacidade de compressão dos dados, mas também à qualidade da sequência a ser transmitida.

A capacidade de compressão de um determinado codificador está relacionada à redução no volume de dados promovida pelo mesmo e, desta forma, pode ser mensurada comparando o valor da taxa de bits (*bitrate*) da sequência de vídeo comprimida com o valor do *bitrate* da sequência original (GHANBARI, 2003). O *bitrate* é uma métrica que expressa a quantidade de dados (bits) transmitida por segundo.

Por outro lado, após passar pelo processo de codificação a qualidade do vídeo é, tipicamente, degradada se comparada ao vídeo original. Considerando que os vídeos digitais são produzidos para serem assistidos por pessoas, a avaliação de qualidade subjetiva, medida a partir da avaliação de satisfação expectadores humanos, é a maneira mais confiável de mensurar a qualidade de um vídeo codificado.

Entre os principais documentos regulamentadores dos testes subjetivos estão as normas BT.500 (ITU-R, 2012) e P9.10 (ITU-T, 1999). A primeira trata da metodologia para avaliação subjetiva de qualidade de imagens de televisão, já a segunda descreve os métodos subjetivos de avaliação de qualidade de vídeo para aplicativo multimídia. Entre eles está o método de avaliação subjetiva DCR (*Degradation Category Rating*), no qual as sequências de teste devem ser apresentadas em pares: a primeira sequência apresentada é sempre a referência e a segunda é a sequência que está sob avaliação. Após analisar, o avaliador deve classificar a degradação de qualidade do vídeo baseado nas seguintes categorias: imperceptível, pouco perceptível, moderadamente perceptível, perceptível, muito perceptível. Este método foi utilizado neste trabalho para a avaliação subjetiva de qualidade do controlador de complexidade proposto, como pode ser visto no Capítulo 7.

Porém, o teste subjetivo não pode ser aplicado durante o processo de codificação. Neste caso, a avaliação de qualidade deve ser realizada a partir de testes objetivos de

qualidade. O uso de critérios objetivos torna este tipo de teste replicável e é indispensável para o processo de otimização taxa-distorção durante o processo de codificação. Além disso, por ser baseado apenas em cálculos matemáticos, este tipo de avaliação é mais fácil e rápido de ser aplicado. Entretanto, ele não é capaz de substituir o teste subjetivo, ou seja, não há sistema de medição objetivo que reproduza a experiência de um observador assistindo o vídeo (AKRAMULLAH, 2014).

A métrica mais popular para calcular a qualidade objetiva de um vídeo é o PSNR (*Peak Signal to Noise Ration*) (RICHARDSON, 2002). O PSNR pode ser utilizado para medir a qualidade objetiva de um bloco, um quadro ou uma sequência de vídeo. Quanto maior o valor desta métrica, maior é a qualidade objetiva. Esta medida logarítmica, depende do número de *bits* por amostra, n, e do critério de distorção conhecido como erro médio quadrático MSE (*Mean Squared Error*), conforme mostra a Equação 2.

$$\mathrm{PSNR}_{db} = 10log_{10}\frac{(2^n-1)^2}{\mathrm{MSE}} \tag{2}$$

O MSE, Equação 3, mede a diferença entre um bloco ou quadro original e o reconstruído a partir do erro médio quadrático das amostras. As variáveis M e N representam, respectivamente, a largura e a altura do quadro. Os termos $f(i,j)$ e $f^{'}(i,j)$ correspondem, ao valor do pixel na localização (i,j) da imagem de origem e na imagem reconstruída, respectivamente.

$$\mathrm{MSE} = \frac{1}{M.N}\sum_{i=1}^{M}\sum_{j=1}^{N}(f(i,j)) - f'(i,j))^2 \tag{3}$$

O formato de vídeo AV1, foco deste trabalho, utiliza a soma das diferenças absolutas SAD (*Sum of Absolute Differences*) e erro quadrático da soma SSE (*Sum Square Error*) como métricas para efetuar a estimativa de movimento (HAN et al., 2021). Essas métricas são definidas pela Equação 4 e Equação 5, respectivamente.

$$\mathrm{SAD} = \sum_{i=1}^{M}\sum_{j=1}^{N}|f(i,j)) - f'(i,j))| \tag{4}$$

$$\mathrm{SSE} = \sum_{i=1}^{M}\sum_{j=1}^{N}(f(i,j)) - f'(i,j))^2 \tag{5}$$

Já o SATD (*Sum of Absolute Transformed Differences*), definido pela Equação 6, é aplicado no software de referência do AV1, Libaom, para realizar uma estimativa rápida

do melhor parâmetro CfL (*Chroma from Luma*) (AOMEDIA, 2016). Os coeficientes $HT(i,j)$ são obtidos a partir da Transformada de Hadamard 2-D de um bloco residual, a qual implica em um maior custo computacional para o SATD se comparado ao SAD. A quantidade de adições necessárias para o cálculo SATD aplicado à uma matriz 4×4 é 5,2 vezes maior que no cálculo do SAD (SOARES et al., 2018).

$$\mathrm{SATD} = \sum_{i=1}^{M}\sum_{j=1}^{N}\left|HT_{(i,j)}\right| \tag{6}$$

Outra métrica, comumente utilizada para medir a eficiência de codificação objetiva de um codificador em relação a um codificador de referência, é o BD *Bjøntegaard Difference* (BD) (BJØNTEGAARD, 2001). Quando aplicada com base no PSNR, é denominada *Peak Signal-to-Noise Ratio* (BD-PSNR). Caso seja obtida a partir do *bitrate*, é conhecida como BD-Rate ou BD-BR. Cabe destacar que a aplicação simultânea delas não é necessária, pois ambas contemplam o mesmo objetivo: mensurar a eficiência de codificação. Considerando que a comparação de *bitrate* e PSNR, isoladamente, não viabiliza nenhuma conclusão em termos de eficiência de codificação, o BD-Rate será usado para avaliar o desempenho de codificação das soluções desenvolvidas ao longo desta tese.

Para obter o BD-Rate, um polinômio logaritmo de terceira ordem é aplicado para gerar duas curvas taxa-distorção (R-D) (BJØNTEGAARD, 2001), uma para o vídeo original e outra para o vídeo comprimido. Como exemplifica a Figura 3, em geral, cada uma das curvas é composta por quatro valores de *bitrate* e quatro valores de PNSR, obtidos através de quatro passos de quantização distintos. Depois, a área entre essas curvas é integrada utilizando o eixo x como referência. O BD-Rate é expresso de maneira percentual. Resultados negativos desta métrica indicam que houve ganho de compressão, isto é, o *bitrate* do vídeo comprimido é menor que o do vídeo original. Por outro lado, resultados positivos significam que, para a mesma qualidade objetiva, o *bitrate* do vídeo comprimido é maior que do vídeo original, em outras palavras, houve perda de compressão.

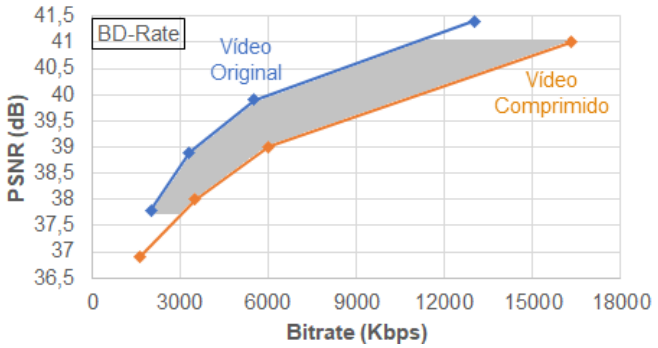


Figura 3 – Curvas de BD-BR para uma dada aplicação.

3 FORMATO DE VÍDEO AV1

Os codificadores de vídeo atuais, incluindo o AV1, utilizam um modelo híbrido, baseado na codificação dos resíduos oriundos da predição em nível de blocos. O fluxo de codificação de vídeo é aplicado em uma estrutura de particionamento de quadro que varia de acordo com o padrão de codificação. Neste capítulo são abordados a estrutura de particionamento AV1 e o fluxo geral de codificação do respectivo controlador.

3.1 Estrutura de Particionamento do AV1

No processo de codificação o quadro é particionado em blocos, cujos tamanhos e formatos podem variar de acordo com o codificador. A flexibilidade do particionamento influencia diretamente no resultado da predição. Quanto maior a quantidade de blocos disponíveis, maior a possibilidade de se obter uma predição otimizada (os resíduos diminuem) e maior a complexidade computacional envolvida. Assim como os demais codificadores, o AV1 divide o quadro a ser codificado em partes menores, possibilitando o tratamento mais adequado à redundância em questão (temporal ou espacial).

O AV1 apresenta uma estrutura de particionamento em árvore que comporta seis níveis de profundidade, como mostra a Figura 4. O tamanho máximo de bloco, denominado SB (*Super Block*), é definido em função das amostras de luminância e pode assumir os tamanhos de 128×128 ou 64×64 amostras. No Libaom (principal software de referência do AV1), o SB do AV1 é definido com o tamanho 128×128, ou seja, são 128 amostras de largura e 128 amostras de altura. Assim, considerando que a raiz da árvore de particionamento criada seja o SB padrão, o bloco quadrático 128×128 é dividido, recursivamente, em quatro partes menores, podendo atingir folhas com tamanho de 4×4 para amostras de luminância (HAN et al., 2021).

O codificador AV1 apresenta uma quantidade considerável de opções de particionamento, como pode ser observado na Figura 5. Os blocos podem ser simétricos e assimétricos, sendo constituídos por diferentes formas geométricas: quadrados ou

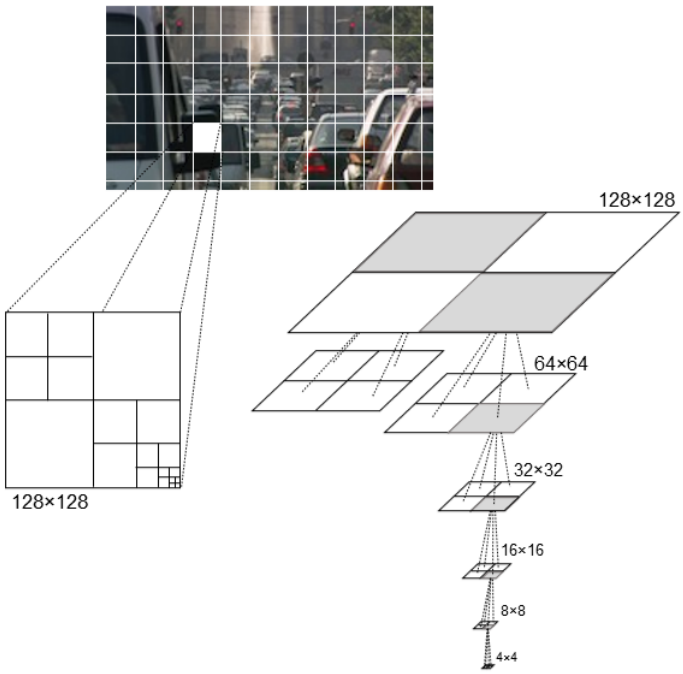


Figura 4 – Níveis de profundidade do particionamento de blocos no AV1.

retângulos. No particionamento denominado NONE, o codificador não divide o bloco, mantendo a razão 1:1 do tamanho quadrático atual. Também é possível particionar o bloco em duas partes, usando razão 1:2/2:1 (HORZ ou VERT) ou em três partes (HORZ_A, HORZ_B, VERT_A e VERT_B) composto por dois blocos quadráticos e um bloco retangular. Por exemplo, o particionamento de um bloco 64×64 usando o modo HORZ_A, resulta na divisão do bloco atual em dois blocos quadráticos 32×32 e um bloco retangular 32×64. O AV1 também permite que o bloco seja particionado em quatro partes (VERT_4, HORZ_4) na razão 1:4/4:1, inovando em relação aos padrões anteriores (CHEN et al., 2020).

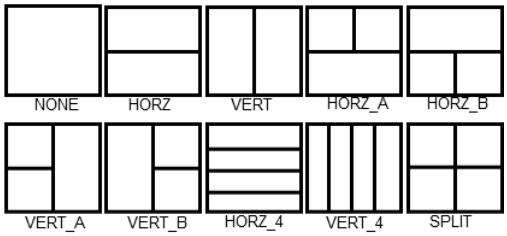


Figura 5 – Possibilidades de formatos de particionamento do codificador AV1. Adaptado de (GU; WEN, 2019).

Além das possibilidades já citadas, existem os blocos divididos em quatro quadrados de tamanhos idênticos (SPLIT). Esse tipo de bloco corresponde à raiz de uma subárvore em cada um dos níveis de codificação. Em cada nível de bloco dentro da estrutura de predição, podem ser usadas até 10 opções diferentes de particionamento. Os blocos quadráticos podem ser divididos de forma recursiva, onde o tamanho mínimo de bloco definido pelo codificador é a condição de parada. As nove opções

de particionamento não quadráticas não suportam subdivisões, tornando-se folhas da árvore de particionamento (CHEN et al., 2020).

A predição inter-quadros suporta os 22 tamanhos de blocos para amostras de luminância possíveis no AV1, como apresentado na Tabela 2. Já a predição intra-quadro e a etapa de transformadas dispõem de um número um pouco menor de opções: são 19 tamanhos diferentes de blocos disponíveis, como apresentado na Tabela 3.

Tabela 2 – Relação dos tamanhos dos blocos existentes no AV1.

Tipo de bloco	**Tamanhos dos blocos**
Quadráticos	128×128, 64×64, 32×32, 16×16, 8×8, 4×4
Retangulares 1:2/2:1	128×64, 64×128, 64×32, 32×64, 32×16, 16×32, 16×8, 8×16, 8×4, 4×8
Retangulares 1:4/4:1	64×16, 16×64, 32×8, 8×32, 16×4, 4×16

Tabela 3 – Relação dos tamanhos dos blocos suportados na predição intra-quadro e etapa de transformadas.

Tipo de bloco	**Tamanhos dos blocos**
Quadráticos	64×64, 32×32, 16×16, 8×8, 4×4
Retangulares 1:2/2:1	64×32, 32×64, 32×16, 16×32, 16×8, 8×16, 8×4, 4×8
Retangulares 1:4/4:1	64×16, 16×64, 32×8, 8×32, 16×4, 4×16

A etapa de transformadas possui uma árvore de particionamento com algumas particularidades. As regras de particionamento de transformada aplicadas a blocos preditos com predição inter-quadros e intra-quadro podem ser observadas nas Figuras 6 e 7, respectivamente. Na predição inter-quadros, cada ramo da árvore pode seguir o particionamento isoladamente e para componentes de luminância, a árvore de transformada pode ter até dois níveis. Independentemente do tipo de predição aplicada ao bloco (inter ou intra), seu tamanho de transformação é o mesmo tamanho de bloco predito, exceto para blocos maiores que 64x64. Neste caso, o bloco de transformação inicial é 64×64. Além disso, o modo SPLIT não é aplicado, exclusivamente, em blocos quadráticos, como ocorre na etapa de predição (HAN et al., 2021).

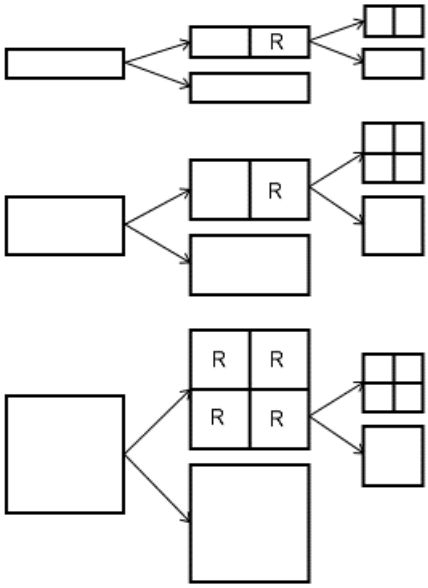


Figura 6 – Particionamento dos blocos de transformada para blocos preditos com predição inter-quadros. Adaptado de (HAN et al., 2021).

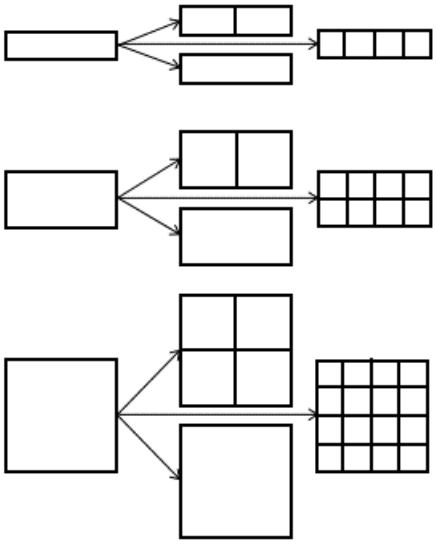


Figura 7 – Particionamento dos blocos de transformada para blocos preditos com predição intra-quadro.Adaptado de (HAN et al., 2021).

3.2 Fluxo Geral de Codificação do AV1

O fluxo da codificação do AV1 é representado na Figura 8 e segue o modelo de diversos outros codificadores atuais. A codificação inicia com a etapa de predição, realizada a partir dos módulos denominados predição intra-quadro e predição inter-quadros. Em seguida, é realizada a codificação residual, por meio dos módulos de transformada (T) e quantização (Q). Por fim, a codificação de entropia é aplicada. Além disso, a decodificação residual, constituída pelas etapas de transformada inversa (T^{-1}), quantização inversa (Q^{-1}), é aplicada, seguida pela etapa de filtragem, a fim de reconstruir o quadro atual para que esse possa ser usado como referência na

codificação dos próximos quadros. Essa operação é realizada para garantir que codificador e decodificador usem exatamente as mesmas referências (AKRAMULLAH, 2014).

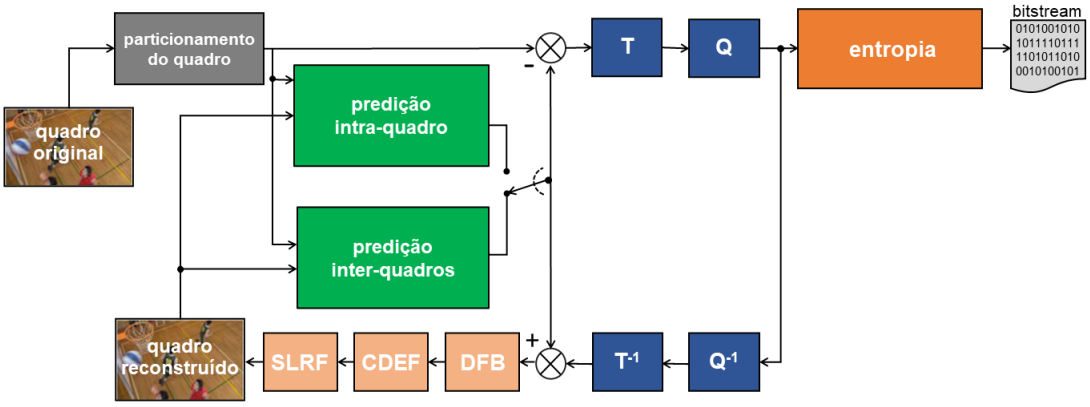


Figura 8 – Modelo simplificado do codificador AV1.

3.2.1 Predição Intra-quadro

A predição intra-quadro, é responsável pela identificação da redundância espacial, contribuindo para a redução no volume de dados, exclusivamente, através do conhecimento dos blocos vizinhos ao bloco a ser predito (RICHARDSON, 2002). O fato de usar apenas informações contidas no próprio quadro para realizar a codificação dos blocos torna a aplicação deste tipo de predição indispensável, por exemplo, no primeiro quadro do vídeo, pois neste momento da codificação ainda não há informações de blocos de quadros anteriores. A predição intra-quadro do AV1 foi uma das etapas de codificação que mais apresentou melhorias e inovações quando comparado a outros codificadores. Além dos 56 modos direcionais, foram introduzidos no AV1 os seguintes modos: *Smooth*, *Paeth*,*Recursive-based-filtering*, *Chroma From Luma* (CfL), *Intrablock Copy* (IntraBC) e *Color Palette* (HAN et al., 2021).

A predição direcional do AV1 é similar às que podem ser encontradas nos codificadores anteriores, como VP9 e HEVC, no entanto, conta com um número bem mais significativo de ângulos, como mostra a Figura 9. Os oito modos direcionais do VP9, usados como base, podem ser variados em até três passos no sentido horário ou no sentido anti-horário, como mostra a Figura 9, sendo que cada passo corresponde a 3°. Dessa forma, esta predição comporta ângulos entre 36 e 212 graus (56 modos).

O modo não-direcional *Smooth*, aplicado em regiões com textura gradiente, é usado para gerar superfícies lisas. Ele realiza a predição baseado na interpolação quadrática de amostras com 1/256 de precisão, com coeficientes aplicados de acordo com a localização do bloco a ser predito. Os três preditores que compõe este modo são: *Smooth*, *Smooth Vertical*, *Smooth Horizontal*, como mostrado na Figura 10. Nos dois primeiros a predição é realizada, respectivamente, a partir da amostra localizadas

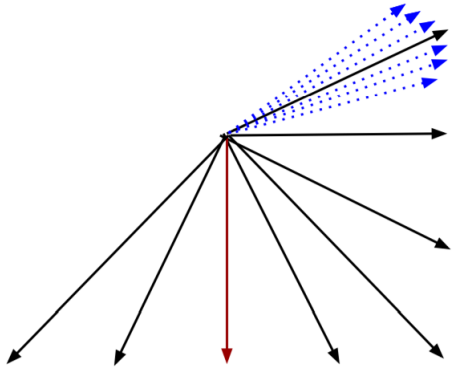


Figura 9 – Modo Direcional do AV1. Fonte: (HAN et al., 2021).

verticalmente e horizontalmente à amostra a ser predita. Já o terceiro realiza a predição considerando a média das duas predições citadas anteriormente. Assim como *Smooth*, o modo *Paeth* também realiza suavização da superfície, porém os blocos preditos usam apenas cópias exatas das amostras de referência.

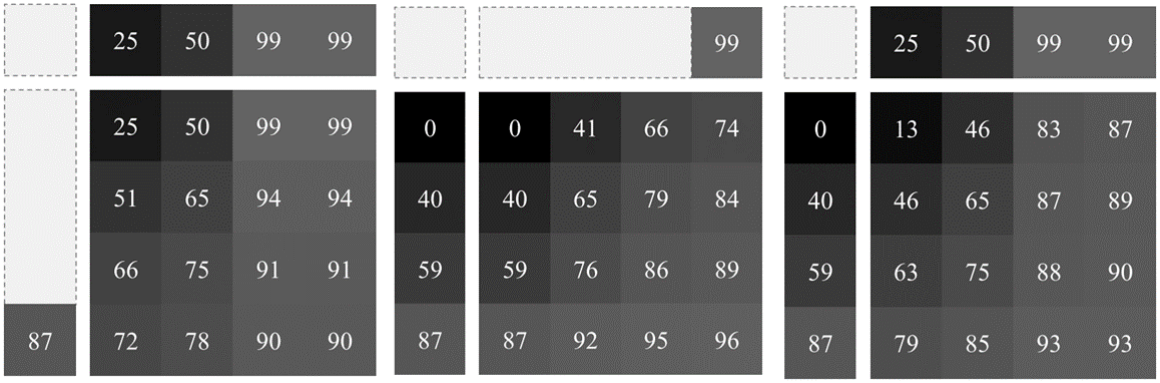


Figura 10 – Bloco 4×4 predito a partir dos preditores *Smooth Vertical*, *Smooth Horizontal* e *Smooth* do AV1. Fonte: (CORRÊA et al., 2020)

O preditor CfL, uma das inovações apresentada pelo AV1, modela os pixels de crominância como uma função linear dos pixels da luminância reconstruídos, baseado na existência de alguma correlação entre os canais da imagem, mesmo após o processo de separação (CHEN et al., 2018). Além disso, o AV1 também possui o modo *Recursive-based-filtering*. Aplicável em amostras de luminância, este modo usa um conjunto de filtros lineares para realizar a predição, explorando a correlação entre os pixels em um bloco 4×2 e sete vizinhos adjacentes a ele (CHEN et al., 2018).

Já o modo *Color Palette* é relevante, principalmente, para vídeos de conteúdo de tela, como por exemplo jogos. Este modo realiza a predição do bloco com base em uma paleta de cores que pode variar de duas a oito cores de base. Para cada canal das amostras, é disponibilizado um conjunto de paletas. Após processadas pelo codificador, tanto as paletas quanto a posição das amostras nas mesmas são enviadas para o decodificar. Existe, também, a possibilidade de utilizar o intraBC, modo que

permite que o codificador intra-quadro faça referência a blocos previamente reconstruídos no mesmo quadro, de maneira similar com que o codificador inter-quadros faz referência a blocos de quadros previamente processados. A aplicação desta técnica tende a apresentar bons resultados quando o vídeo contém texturas repetidas, como é comum em vídeos de conteúdo de tela.

3.2.2 Predição Inter-quadros

A predição inter-quadros visa identificar a redundância temporal existente entre os quadros temporalmente vizinhos de uma cena. Devido às altas taxas de amostragem da captação dos vídeos, conforme mencionado no Capítulo 2, a redundância temporal é geralmente mais representativa que a redundância espacial.

A redundância temporal entre os quadros é encontrada a partir das técnicas de Estimação de Movimento (ME – *Motion Estimation*) e Compensação de Movimento (MC – *Motion Compensation*). A ME, etapa que apresenta a maior complexidade computacional do codificador (PURI; CHEN; LUTHRAC, 2004), inicia com a escolha do(s) quadro(s) de referência. A área de busca é constituída por uma quantidade limitada de blocos candidatos existente no quadro de referência e a escolha do bloco candidato pode ser feita por diversos algoritmos de busca (PORTO, 2012). Os blocos do quadro atual são comparados com os blocos do quadro de referência a partir de critérios de similaridade, como, por exemplo, MSE ou SAD (KUHN, 1999). Os critérios que geram maior similaridade e, consequentemente, menor resíduo são escolhidos para a predição. Para cada bloco do quadro atual é gerado um vetor de movimento, com duas direções, que corresponde ao deslocamento do bloco predito em relação ao quadro de referência (RICHARDSON, 2002). Os vetores de movimento também são utilizados pela etapa de MC, a qual é responsável pela reconstrução dos quadros preditos.

Na predição inter-quadros, por padrão, o AV1 utiliza um GFG (Golden Frame Group) com estrutura de multicamadas, conforme mostrado na Figura 11. Esta técnica de agrupamento realiza o processamento de determinados tipos de quadros de maneira específica. Os GFGs são dinâmicos, podendo variar a quantidade de quadros que os compõem, sendo que a quantidade máxima de quadros permitida é 16 (CHEN et al., 2020). É possível avaliar de três a sete tipos de quadros de referência, são eles: LAST (vizinho passado mais próximo), LAST2 (segundo vizinho passado), LAST3 (terceiro vizinho passado), GOLDEN (quadro passado possível de ser usado diversas vezes), ALTREF, ALTREF2 e BWDREF. O ALTREF é uma referência futura construída a partir de filtragem temporal ao longo da trajetória de movimento de quadros originais consecutivos. O ALTREF2 corresponde a referência obtida a partir de mais de um candidato ALTREF dentro do GFG. O BWDREF é um quadro de referência futuro obtido sem nenhum tipo de filtragem.

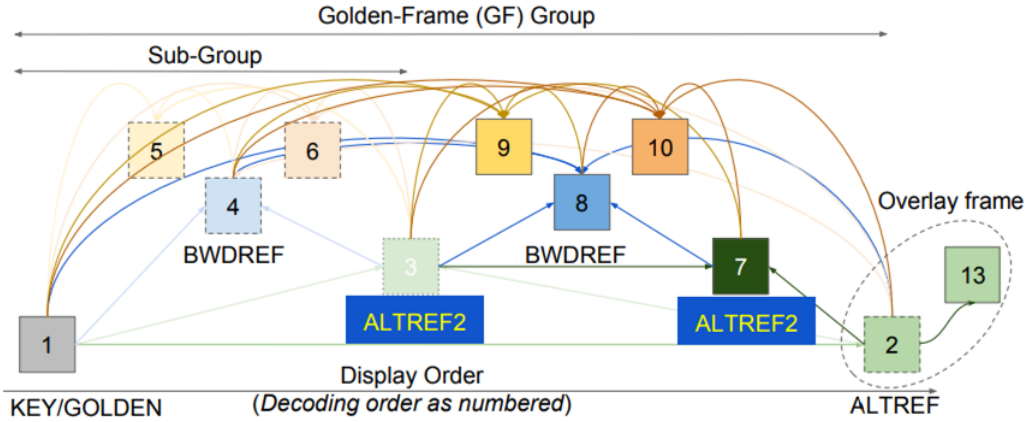


Figura 11 – Estrtutura de um GFG. Fonte: (CHEN et al., 2020).

A predição de cada bloco pode ser realizada usando um único quadro de referência, através do modo SRFPM (*Single Reference Frame Prediction Mode*) ou combinando predições de dois quadros de referência, com o modo CRFPM (*Compound Reference Frame Prediction Mode*). Para o modo SRFPM, existem quatro vetores de movimento candidatos: NEARESTMV, NEARMV, NEWMV e GLOBALMV. Os vetores NEARESTMV e NEARMV correspondem, respectivamente, ao vetor de movimento mais provável e ao segundo vetor mais provável entre os vetores de movimentos codificados anteriormente na vizinhança da unidade de predição atual, conforme ilustrado na Figura 12. Os vetores NEWMV e GLOBALMV correspondem, respectivamente, a aplicação de um novo vetor no fluxo de codificação e um único vetor de movimento para todo o quadro. Já no modo CRFPM, o número de vetores de movimento possíveis é estendido para oito. Eles são obtidos a partir da combinação de alguns dos candidatos do modo SRFPM.

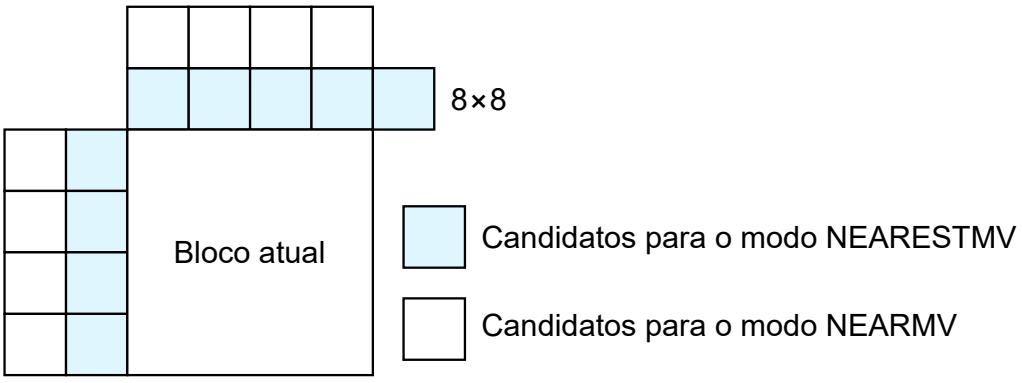


Figura 12 – Modos NEARESTMV e NEARMV. Adapatado de (KIM et al., 2019).

A predição composta pode adotar os modos: *Average Predictor*, *Distance Weighted Predictor*, *Difference Weighted Predictor* e *Wedge Mode*. Esses modos diferem, basicamente, em relação ao peso atribuído a cada preditor. No caso mais simples, *Average Predictor*, as duas referências são ponderadas igualmente. No modo *Distance Weighted Predictor* o peso de cada preditor é definido com base na sua dis-

tância temporal em relação ao quadro atual. No modo *Difference Weighted Predictor* o coeficiente de ponderação é calculado por pixel a partir da diferença entre os dois pixels de referência. Por fim, no modo *Wedge Mode* a predição é realizada com base em 16 possibilidades de máscaras predefinidas. O bloco é divido em duas partes de acordo com a máscara selecionada e cada parte é preenchida com os pixels de um bloco de referência. A aplicação dos três últimos modos é demonstrada na Figura 13.

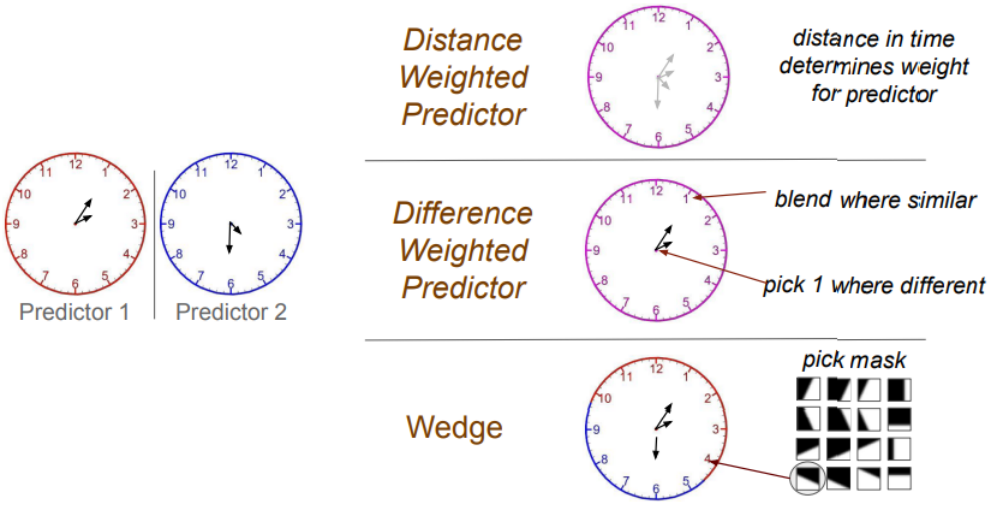


Figura 13 – Ilustração dos modos de predição composta. Fonte: (HAN et al., 2021).

Outro modo composto é o *Compound Inter-Intra*. Ele permite combinar a predição intra-quadro e predição inter-quadros de duas maneiras: usando *wedge*, semelhante ao que ocorre na predição inter-inter ou *smooth*. Também há a técnica denominada OBMC (*Overlapped Block Motion Compensation*), que reduz erros de predição nas bordas, combinando suavemente predições criadas a partir de vetores de movimento de blocos vizinhos adjacentes (CHEN et al., 2018).

O codificador AV1 também explora modelos de compensação de movimento *warped*, ou seja, modelos destinados à predição de movimentos complexos que requerem, por exemplo, transformações no dimensionamento, rotações e alteração de proporções. Esses movimentos podem ser gerados em casos específicos como trepidação da câmera, movimentos panorâmicos e zoom. Segundo Parker et al. (2017), há dois modos de predição *warped*: *Local Warped Motion* e *Global Motion*. Além disso, o AV1 usa a estimação de movimento fracionária, ainda mais precisa que o VP9, composta por 90 filtros diferentes que possuem de 2 a 8 taps. São quatro famílias de filtros: *normal*, *smooth*, *sharp* e *bilinear* (AOMEDIA, 2018b). A precisão é de um oitavo de amostra para luminância e um dezesseis avos de amostra para as amostras de crominância (HAN et al., 2021).

3.2.3 Transformadas

A etapa de predição gera como resultado os resíduos, obtidos através da diferença entre o quadro reconstruído, de acordo com as predições, e o quadro original. A etapa seguinte, Transformada (T), tem como principal objetivo converter esses dados residuais, que estão no domínio espacial, para o domínio das frequências. Esta transformação de domínio auxilia a identificação das informações de alta frequência, que são menos relevantes para o sistema visual humano, e poderão ser melhor exploradas nas próximas etapas do processo de codificação (RICHARDSON, 2002). As transformadas podem ser aplicadas em diferentes tamanhos e formatos de bloco.

Além da significativa quantidade de blocos, já mencionada na Seção 3.1, o AV1 usa um conjunto muito rico de diferentes transformadas. São quatro tipo de transformadas, DCT (*Discrete Cosine Transform*), ADST (*Asymmetrical Discrete Sine Transform*), flipADST e IDTX (*Identity Transform*) (PARKER et al., 2016) que podem ser aplicadas nas amostras residuais, ou ainda, combinadas com uma transformada na direção horizontal e outra na direção vertical, totalizando dezesseis combinações possíveis. Conforme apresentado na Tabela 4, o AV1 utiliza seis diferentes conjuntos de transformadas.

Um deles utiliza apenas a transformada DCT (DCTONLY). Também existem dois conjuntos específicos para a predição intra-quadro: INTRA1 e INTRA2. O INTRA1 é composto pelas transformadas DCT-DCT, ASDT-DCT, DCT-ADST, ADST-ADST, IDTX, V-DCT e H-DCT. Já o INTRA2 é constituído pelas mesmas transformadas do conjunto anterior, com exceção das transformadas V-DCT e H-DCT. Os outros três conjuntos restantes são referentes à predição inter-quadros, são eles: INTER1, INTER2 e INTER3. O O conjunto INTER1 contempla as 16 possibilidades de transformadas. O INTER2 é similar ao INTER1, porém não conta com as transformadas V-ADST, H-ADST, V-FLIPADST e H-FLIPADST. Por fim, o INTER3, conjunto de transformadas mais reduzido referente à predição inter-quadros, utiliza apenas as transformadas DCT-DCT e IDTX.

A configuração padrão permite que cada quadro, de acordo com o tipo de predição e o tamanho do bloco de transformação, seja codificado a partir de um dos conjuntos descritos. Existe também a possibilidade de limitar o codificador a determinados conjuntos, como apresentado no Apêndice A.

3.2.4 Quantização e Entropia

Após a etapa de transformadas, etapa de quantização (Q), explora a característica de perda de sensibilidade do sistema visual humano para informações de alta frequência, atenuando informações menos perceptíveis ao olho humano. Esta etapa atenua os coeficientes transformados com maior ou menor intensidade de acordo com a sua característica (alta ou baixa frequência), a partir do ajuste do passo de quantização,

Tabela 4 – Conjuntos de transformadas do codificador AV1.

Tipo	DCTONLY	INTRA1	INTRA2	INTER1	INTER2	INTER3
DCT-DCT	X	X	X	X	X	X
ASDT-DCT		X	X	X	X	
DCT-ADST		X	X	X	X	
ADST-ADST		X	X	X	X	
FLIPADST-DCT				X	X	
DCT-FLIPADST				X	X	
FLIPADST-FLIPADST				X	X	
ADST-FLIPADST				X	X	
FLIPADST-ADST				X	X	
IDTX		X	X	X	X	X
V-DCT		X		X	X	
H-DCT		X		X	X	
V-ADST				X		
H-ADST				X		
V-FLIPADST				X		
H-FLIPADST				X		

gerando matrizes esparsas e potencializando a codificação de entropia. As perdas inseridas e a taxa de compressão são diretamente proporcionais aos valores aplicados ao passo de quantização, ou seja, quanto maior o passo de quantização, maiores são as perdas inseridas e maiores são as taxas de compressão (RICHARDSON, 2002).

O passo de quantização, chamado no AV1 de CQ (*Constant Quality*), pode assumir valores entre 0 e 63. As perdas inseridas e a taxa de compressão são diretamente proporcionais aos valores de CQ aplicados, sendo que quanto maior o valor do CQ, maiores serão as perdas de informação e, consequentemente, maior será o potencial de compressão e de perdas na qualidade do vídeo codificado.

A etapa de codificação de entropia é aplicada nas amostras residuais após a transformada e quantização, gerando o *bitstream* que será interpretado pelo decodificador. Além dos resíduos, esta etapa também codifica as informações laterais, como modos de predição e vetores de movimento, por exemplo.

O codificador de entropia do AV1 é composto por um codificador aritmético multi-símbolos (HAN et al., 2021), que permite que vários símbolos binários sejam combinados em símbolos não binários. Seu alfabeto é constituído por 16 símbolos, codificados com probabilidades de até 15 bits. Este codificador de entropia apresenta maior eficiência em relação ao codificador aritmético binário (usado no HEVC e no VVC, por exemplo) (LAUDE et al., 2018).

3.2.5 Filtros

Para a reconstrução do quadro atual, o qual, poderá ser usado como referência para o próximo quadro a ser codificado pela predição inter-quadros, são aplicadas as

etapas de Q^{-1} e T^{-1}, além da MC. A finalização da reconstrução deste quadro ainda passa por uma etapa de filtragem, que é responsável pela suavização dos artefatos inseridos ao decorrer do processo de codificação (RICHARDSON, 2002), principalmente pelas etapas de predição e quantização.

O AV1 possui três filtros de laço principais: o DBF o (*Deblocking Filter*), o CDEF (*Constrained Directional Enhancement Filter*) e o SLRF (*Switchable Loop Restoration Filter*). Estes filtros, presentes tanto no codificador quanto no decodificador, eliminam ou reduzem efeitos de blocos, *ringing* e *blurring*, respectivamente (CARVALHO NOBRE PALAU, 2022) Os efeitos de blocos são gerados pelo fato que diferentes blocos podem ser codificados independentes de seus blocos vizinhos e com ferramentas distintas. A inconsistência entre esses blocos resulta em descontinuidades e artefatos nas bordas dos blocos que acabam degradando a qualidade da imagem. O *ringing* é causado pela etapa de quantização, que tende a eliminar as frequências mais elevadas da imagem, causando uma espécie de granulação nas arestas de alta frequência da imagem. Já o *blurring* é a desfocagem que acontece durante o processo de codificação, tipicamente, produzida pelo uso do filtro para reduzir o efeito de bloco (HSU; SHEN, 2017).

3.3 Software de Referência do AV1 - Libaom

O formato de codificação vídeo AV1 está incluso em diferentes codificadores e decodificadores, como o codificador/decodificador SVT-AV1 e o codificador Rav1e (XIPH, 2020), ambos desenvolvidos por parceiros da AOMedia. No entanto, a principal implementação do codificador AV1 é o Libaom (AOMEDIA, 2018a), o qual foi desenvolvido pela própria AOMedia e é considerado o software de referência do codificador AV1. O Libaom implementa todas as ferramentas disponíveis no formato de codificação AV1, sendo a principal implementação utilizada pelos pesquisadores que trabalham com o AV1.

A configuração padrão do Libaom usa duas passadas no processo de codificação, ou seja, duas etapas. Na primeira passada são coletados dados estatísticos do vídeo a ser codificado. Posteriormente, na segunda passada, essas informações são usadas para a otimização e/ou aceleração de diferentes etapas do processo de codificação para que a compressão do vídeo de entrada seja, de fato, executada (CHEN et al., 2018). Na primeira passada, por exemplo, o AV1 define, a partir de blocos quadráticos, os limites mínimos e máximos da árvore de particionamento, os quais são usados pela segunda passada durante o processo de codificação (CHIANG; HAN; XU, 2019).

3.4 Comparação entre AV1 e VP9

O AV1 foi desenvolvido para atender as demandas do mercado atual de vídeo digital. Para alcançar este objetivo, algumas ferramentas foram desenvolvidas especificamente para este codificador e outras foram aperfeiçoadas de codificadores já existentes, principalmente, do VP9, seu predecessor. Uma breve comparação entre as características do AV1 e VP9 é apresentada na Tabela 5, permitindo estimar a alta complexidade envolvida no processo de codificação do AV1.

Uma expressiva novidade implementada no AV1 em relação ao seu antecessor está no particionamento. ambos codificadores permitem subdividir o tamanho máximo de bloco, recursivamente, em blocos de até 4×4 amostras. No entanto, o AV1 define o tamanho máximo de bloco como 128×128 e o VP9 como 64×64. Além disso, no codificador AV1, é possível usufruir de mais seis opções de particionamento que o VP9, aplicados tanto à predição intra-quadro quanto à predição inter-quadros (HAN et al., 2021) (MUKHERJEE et al., 2013).

A predição do AV1 também é, notavelmente, mais complexa que a do seu antecessor. Sua predição intra-quadro possui 59 modos a mais que o VP9, sendo 48 novos modos direcionais e 11 novos modos não-direcionais. Na predição inter-quadros é possível avaliar até quatro quadros de referência a mais que no VP9. Além disso, na predição composta do AV1 os quadros podem ser combinados de 16 maneiras diferentes, já no VP9 apenas cinco combinações são permitidas (HAN et al., 2021) (MUKHERJEE et al., 2013).

Na etapa de transformada também podem ser observadas melhorias significativas. São definidos 19 tamanhos de blocos no AV1 e quatro tipos de transformadas (DCT, ADST, flipADST, IDTX), contra quatro tamanhos de blocos de transformadas e três tipos de transformadas (DCT, ADST, WHT) permitidos no VP9. A quantidade de combinações de transformadas possíveis no AV1 é o dobro que a permitida no VP9, 16 contra oito (HAN et al., 2021) (MUKHERJEE et al., 2013).

Já na etapa de codificação de entropia, o AV1 utiliza um codificador aritmético multi-símbolo adaptativo, enquanto o VP9 emprega um codificador aritmético binário, não adaptativo, baseado em árvore para codificar todos os elementos de sintaxe. O codificador AV1 também dispõe de três filtros distintos para reconstruir os quadros preditos, enquanto o VP9 usa apenas um (CARVALHO NOBRE PALAU, 2022) (MUKHERJEE et al., 2013).

Esta breve comparação ajuda a visualizar algumas melhorias apresentadas pelo AV1 em relação ao VP9 em diferentes etapas de codificação. As ferramentas adicionadas ou otimizadas contribuem para o aumento da eficiência de codificação, mas, em contrapartida, promovem importante acréscimo na complexidade e, consequentemente, no tempo de processamento (como apresentado no Capítulo 1) e no consumo

de energia, fatores que, na prática, prejudicam a aplicação do AV1, especialmente em sistemas embarcados, de menor poder de processamento, e em dispositivos móveis alimentados por bateria.

Tabela 5 – Comparação entre os codificadores de vídeo AV1 e VP9.

	AV1	**VP9**
Faixa de tamanho de blocos	4×4 até 128×128	4×4 até 64×64
Tamanhos de blocos permitidos	4×4, 4×8, 8×4, 4×16, 16×4, 8×8, 8×16, 16×8, 8×32, 32×8, 16×16, 16×32, 32×16, 16×64, 64×16, 32×32, 32×64, 64×32, 64×64, 64×128, 128×64, 128×128	4×4, 4×8, 8×4, 8×8, 8×16, 16×8, 16×16, 16×32, 32×16, 32×32, 32×64, 64×32, 64×64
Tipos de particionamento de blocos	Um quadrado, um split, duas divisões binárias, quatro ternárias e duas quaternárias	Um quadrado, um split e duas divisões binárias
Modos Intra Direcionais	56 (de 36º até 212º)	8 (de 45º até 207º)
Modos Intra Não-Direcionais	DC, 3 Smooth, Paeth, 5 Recursivo, CfL, Color Pallete, IntraBC	DC and TM (True Motion)
Quadros de referência Inter	até 7	até 3
Tipos de quadros de referência Inter	LAST_FRAME GOLDEN_FRAME ALTREF_FRAME LAST2_FRAME LAST3_FRAME BWDREF_FRAME ALTREF2_FRAME	LAST_FRAME GOLDEN_FRAME ALTREF_FRAME
Combinações dos quadros de referência	16	5
Tamanho de blocos de transformadas	4×4, 4×8, 8×4, 4×16, 16×4, 8×8, 8×16, 16×8, 8×32, 32×8, 16×16, 16×32, 32×16, 16×64, 64×16, 32×32, 32×64, 64×32, 64×64, 4×4, 8×8, 16×16, 32×32	4×4, 4×8, 8×4, 4×16, 16×4, 8×8, 8×16, 16×8, 8×32, 32×8, 16×16, 16×32, 32×16, 16×64, 64×16, 32×32, 32×64, 64×32, 64×64, 4×4, 8×8, 16×16, 32×32
Tipos de transformadas	DCT, ADST, flipADST, IDTX	DCT, ADST, WHT
Combinações de Transformadas	16	8
Entropia	Huffman, Multi-símbolo	variação do CABAC
Filtros de reconstrução	DBF, CDEF, SLRF	Loop Filtering

4 TRABALHOS RELACIONADOS

O AV1 pode ser considerado um codificador recente se comparado a outros codificadores existentes no mercado. Mesmo que se possa observar na literatura trabalhos voltados para este formato de vídeo, a quantidade de trabalhos ainda é limitada.

Tendo em vista que este trabalho tem como foco o controle dinâmico de complexidade do AV1, artigos científicos que tratam de redução de complexidade e controle de complexidade aplicados a este codificador foram pesquisados. Foi realizada uma breve revisão na literatura, baseada nas seguintes perguntas norteadoras:

- Existem soluções para a redução de complexidade do codificador AV1?
- Existem soluções para o controle de complexidade do codificador AV1?

Todas as pesquisas foram feitas usando o *Google Scholar* (GOOGLE, 2022) e sem limitação do período de busca. Inicialmente, foi realizada uma pesquisa abrangente com base na palavra-chave "AV1 *encoder*". Dentre os resultados, foram encontrados 13 artigos relacionados à redução de complexidade, os quais serão apresentados na Seção 4.1. Não foram encontrados trabalhos referentes à controle de complexidade no AV1. Diante disso, uma nova pesquisa foi realizada a partir da *string* “VP9” e “*complexity control*”. No entanto, nenhum dos resultados atendeu, de fato, à *string* pesquisada. Levando em consideração que este é o tema principal deste trabalho, optou-se, então, por expandir a pesquisa deste assunto para outra família de codificadores.

Assim, usando a mesma base de dados, foram realizadas mais duas pesquisas com o intuito de buscar trabalhos científicos que apresentam soluções para o controle de complexidade do codificador VVC e seu antecessor, o HEVC. Foram usadas as seguintes *strings*: “VVC” e “*complexity control*” e “HEVC” e “*complexity control*”. Foram encontrados dois artigos referentes à controle de complexidade do codificador VVC. Para a segunda busca, além de artigos referentes ao período de 2018 a 2022, foram selecionados os que apresentam número de citações superior a dez, conforme a métrica i10, a qual é adotada pelo Google Scholar desde 2011 (DHAMDHERE, 2018). Ao total são abordados 24 artigos sobre controle de complexidade no HEVC. Eles são apresentados na Seção 4.2.

4.1 Redução de Complexidade no Codificador AV1

Esta seção traz uma breve discussão sobre os 13 trabalhos encontrados na literatura, até o presente momento da escrita desta tese, voltados à redução da complexidade do AV1.

Mais de 60% das publicações encontradas atingem a redução de complexidade explorando a árvore de particionamento. Este valor significativo pode ser explicado devido ao fato do particionamento requerer uma significativa capacidade de processamento e, consequentemente, tempo de codificação.

Em Guo; Han; Wen (2018) é apresentado um algoritmo rápido baseado na codificação de múltiplas resoluções. A similaridade das estruturas de blocos entre as resoluções é explorada para acelerar a codificação das altas resoluções. A decisão rápida da estrutura de blocos das resoluções mais elevadas é realizada com base nos resultados de RDO das baixas resoluções e a profundidade média da vizinhança colocalizada de bloco é usada para definir a terminação antecipada do processo RDO.

Outra solução com o objetivo de melhorar a estrutura de blocos de particionamento, foi proposta por Guo et al. (2018), porém, com base em um modelo de inferência Bayesiana. Posteriormente, o mesmo método também foi usado em Chen et al. (2019), para realizar a decisão dos blocos de particionamento do AV1 baseado na pré-codificação do HEVC.

Em 2019, Chiang; Han; Xu (2019) propuseram uma solução, baseada em dois estágios, para reduzir a complexidade a partir da árvore de particionamento. Na primeira etapa é construída uma árvore de partição binária recursiva na qual apenas blocos quadrados são considerados. O custo RD de cada bloco de codificação é estimado apenas referenciando os quadros de referência mais próximos e o bloco de transformação assume o tamanho máximo disponível para blocos codificados com 2D-DCT. Um subconjunto formado pelos blocos que apresentam melhor custo RD são fornecidos para a etapa seguinte. Na segunda etapa, para cada bloco quadrado oriundo da etapa anterior, o particionamento mais adequado é pesquisado entre as dez estruturas de predição possíveis para o AV1 (CHEN et al., 2018).

A estrutura de particionamento das transformadas também é explorada para reduzir a complexidade do codificador AV1. Na solução proposta por Su et al. (2019) é usado aprendizagem de máquina para modificar a estrutura de particionamento desta etapa de codificação. Os modelos usam recursos de entrada extraídos do bloco residual, como desvio padrão, correlação e distribuição de energia. A rede neural desenvolvida estima a probabilidade dos tamanhos do bloco de transformação e o tipo de transformada, assim, o codificador pode eliminar os tamanhos de blocos e candidatos que, provavelmente, não serão selecionados.

No mesmo ano, Kim et al. (2019) apresenta um método de decisão para realizar o

término antecipado da árvore de particionamento para a predição inter-quadros e avaliar a necessidade de testar todos os modos inter-quadros apresentados pelo AV1. O método, baseado em mineração de dados, define, através de uma classificação binária, se o bloco deve ser predito usando modo de referência simples ou composta. Se o bloco for classificado para ser predito com referência simples, os modos de referência composta não serão testados. Caso contrário, os testes usualmente realizados pelos codificadores AV1 convencionais serão realizados.

Um método para a redução de complexidade da predição inter-quadros do AV1 também foi apresentado em Gu; Wen (2019). Baseado nas informações fornecidas pela profundidade intermediária da árvore de particionamento, o método estima a distribuição probabilística das decisões de particionamento e realiza a remoção rápida de unidades de predição. A decisão de divisão do bloco é realizada a partir do grau de partição dele, que é quantificado através da média ponderada da profundidade absoluta de todas as unidades de predição do respectivo bloco. A mesma informação é usada, posteriormente, para acelerar o processo de seleção do tipo de partição. Mais recentemente, outro algoritmo baseado em árvore de decisão para a predição inter-quadros, prevendo as decisões de divisão em cada profundidade, é proposto por Chen et al. (2021).

Porém, também foi possível encontrar trabalhos que atingiram a redução de complexidade explorando outras etapas de codificação. Em Gankhuyag; Jeong; Kim (2019) é apresentada uma solução voltada à codificação de *streaming* de realidade virtual (360/VR). O método desenvolvido limita a faixa de pesquisa MV e restringe opções de codificação do AV1 não compatíveis a este tipo de *streaming* (desativa o warped_motion, por exemplo). Também é realizada a desativação dos filtros de *loop* nas regiões limitantes dos tiles. Os mesmos pesquisadores propuseram em Jeong; Gankhuyag; Kim (2019) e Jeong; Gankhuyag; Kim (2019a) algoritmos de decisão rápida aplicados aos modos intra-quadro do codificador AV1. Em Jeong; Gankhuyag; Kim (2019), a decisão dos modos de predição intra-quadros explora o RDM (*Rate Distortion Model*). O RDO é executado apenas para os casos que não estão contemplados na margem (fixa) de valores de custo RDM definida pelo codificador. O trabalho desenvolvido mostra que a atualização da margem de forma adaptativa de acordo com a precisão RDM é mais conveniente. A precisão é estimada em tempo real e a margem é definida considerando a variância obtida a partir do custo RDO e do custo RDM. Tanto a precisão do RDM quanto a variância são calculados para cada tamanho de bloco. Em Jeong; Gankhuyag; Kim (2019a), com base em uma análise estatística prévia, os possíveis candidatos de modos de crominância são eliminados a partir do modo de luminância selecionado.

Ainda em 2019, foi proposta a substituição do filtro *in-loop* do AV1 por uma CNN (*Convolutional Neural Network*) foi proposta em (CHEN et al., 2019a). O objetivo prin-

cipal do trabalho é aumentar a eficiência de codificação, no entanto, para a predição inter-quadros os resultados também mostram redução de complexidade. Dois anos depois, uma solução que explora a correlação angular, no modo direcional da predição intra-quadro, entre os componentes luma e croma é proposta por (JIN et al., 2021).

Os trabalhos mencionados acima empregam diferentes abordagens. Há pesquisas que realizam a redução de complexidade do codificador atacando mais de uma etapa de codificação, como é o caso de (CHIANG; HAN; XU, 2019), já outras são voltadas apenas para uma etapa específica da codificação, como por exemplo, (KIM et al., 2019), (SU et al., 2019) e (CHEN et al., 2019a). Cabe ainda destacar a diversidade nos métodos empregados. Alguns trabalhos apresentam soluções heurísticas, como é o caso de Guo; Han; Wen (2018), Gu; Wen (2019) e Chen et al. (2019). Outros baseiam-se em métodos mais complexos, como Kim et al. (2019) e Chen et al. (2019a), que utiliza aprendizado de máquina. No entanto, apesar dos diferentes graus de complexidade, todos contribuem, de alguma forma, na redução da complexidade de codificação.

A Tabela 6 apresenta um panorama geral das principais características e resultados alcançados em relação a redução do tempo de codificação e impacto na eficiência de codificação no AV1 dos trabalhos relacionados até aqui discutidos. Para cada solução é identificada a etapa de codificação explorada para obter a redução de complexidade, assim como a versão do Libaom utilizado para obter os resultados e os respectivos valores médios de redução de tempo de codificação (RTC) e BD-Rate. Cabe destacar que a abreviatura "Part." presente na Tabela 6 corresponde à etapa particionamento.

A RTC obtida variou de 4% a 64,14%, já os valores de BD-Rate encontram-se na faixa de -0,4% a 10%. As três melhores soluções em termos de ganho médio de redução de complexidade foram (CHIANG; HAN; XU, 2019), (KIM et al., 2019) e (GUO; HAN; WEN, 2018), atingindo, respectivamente, RTC de 64,14%, 43,4% e 36,8%, com BD-Rate médio de 0,61%, 0,77% e 1,04%.

4.2 Controle de Complexidade no VVC e HEVC

Até o presente momento, a literatura disponibiliza apenas dois trabalhos sobre o controle de complexidade do VVC, ambos publicados em 2022. Em Huang et al. (2022) a complexidade da predição intra-quadro do VVC é controlada, dinamicamente, a partir da profundidade da árvore QTMT (*Quad-Tree plus Multi-Type Tree*) de cada CTU (*Coding Tree Unit*). A estimativa do tempo de compressão de cada CTU é realizada com base no custo RDO do modo planar da componente de luminância da CTU. A entrada do sistema é um tempo de codificação alvo e o *budget* é realizado a nível

Tabela 6 – Resumo dos trabalhos encontrados na literatura referentes à redução de complexidade no AV1.

Artigo	**Etapa de Codificação**	**Versão Liboam**	**RTC (%)**	**BD-Rate (%)**
(GUO; HAN; WEN, 2018)	Part.	-	36,8	1,04
(GUO et al., 2018)	Part.	f0d710 (7 abr. 2018)	33,4	0,14
(GANKHUYAG; JEONG; KIM, 2019)	Inter	-	17,3	10
(JEONG; GANKHUYAG; KIM, 2019)	Modo Intra	-	8,67	0,04
(JEONG; GANKHUYAG; KIM, 2019a)	Modo Intra	-	15,86	0,44
(KIM et al., 2019)	Part. Inter	d43b65 (3 set. 2018)	43,4	0,77
(SU et al., 2019)	Part. Transf.	d48388 a 5695d7 (1º a 9 jun. 2019)	14,57	0,10
(CHIANG; HAN; XU, 2019)	Part.	-	64,14	0,61
(GU; WEN, 2019)	Part. Inter	3715bf (25 jun. 2018)	29,06	0,95
(CHEN et al., 2019)	Part.	365eae (27 mar. 2018)	35,7	0,61
(CHEN et al., 2019a)	Filtro *in-loop*	514b52 a 1c2752 (3 a 31 jan. 2019)	4,12	4,10
(CHEN et al., 2021)	Part. Inter	d1d122 (18 maio 2020)	23,6	0,73
(JIN et al., 2021)	Modo Intra	299b96 (26 maio 2020)	4	-0,4

de quadro e CTU. Posteriormente, os autores expandiram o controle dinâmico para quadros inter e passaram a controlar a complexidade de cada LCU (*Largest Coding Unit*) do VVC. Um sistema de controle foi implementado que considera um *budget* e *feedback* em diferentes níveis: GOP, quadro e LCU. A pré-alocação de complexidade é realizada com base em modelo linear, custo SATD e ID temporal (HUANG et al., 2022a).

Por outro lado, é possível encontrar literatura diversos trabalhos que exploram controle de complexidade no codificador HEVC. Um controle de complexidade adaptativo que ajusta dinamicamente o codificador aos recursos computacionais através de parâmetros de codificação HEVC, na grande maioria relacionados à predição inter-quadros, foi proposto por Grellert et al. (2013). O controlador, o qual utiliza um circuito de *feedback* PID (Proporcional Integral Diferencial), considera a frequência de operação e a taxa de quadros alvo para calcular um *budget* para cada quadro e assim distribui entre suas CUs (Coding Units). Posteriormente, os autores desenvolveram um modelo que estima a complexidade de codificação com base no número de chamadas de operações básicas de codificação. O controle funciona a nível de quadros e a alocação de orçamento a nível de CTU (GRELLERT et al., 2017).

Uma das alternativas mais exploradas para alcançar a RTC alvo é o ajuste da profundidade máxima das CUs. Em Corrêa et al. (2011), o ajuste da profundidade da CU é realizado de acordo com o tempo de codificação alvo, sem olhar para as profundidades escolhidas nos blocos dos quadros anteriores. Já em Corrêa et al. (2012), Corrêa et al. (2013), Corrêa et al. (2014) e Jimenez-moreno; Martínez-enríquez; María (2016) a profundidade da CU é limitada com base na profundidade máxima do bloco co-localizado do último quadro sem redução de complexidade, na compensação de movimento, na correlação espacial e temporal entre CTUs vizinhas e nas as estatísticas dos custos RD, respectivamente. Em Corrêa et al. (2015) e Corrêa et al. (2016) O ajuste de complexidade computacional é dividido em dois tipos de granularidades: fina e média. Na granularidade fina é aplicado o método proposto em (CORRÊA et al., 2014), enquanto na granularidade média, os pontos operacionais são identificados através de frente de Pareto aplicados a parâmetros de configuração e algoritmos de decisões antecipadas. Quatro anos depois, a eficiência de Pareto também é usada por Hosseini et al. (2020) em um método de controle de complexidade para codificação intra-quadro. O método proposto, ajusta a complexidade através da limitação do número de modos intra-quadro em cada CU. Diferentemente das soluções citadas até aqui, (CHEN et al., 2018a) propõe em controle de complexidade hierárquico, no qual a locação de complexidade é realizada a nível de GOP, quadro e CTU.

O ajuste da profundidade máxima LCU também é explorado em Deng et al. (2016). A alocação de complexidade, realizada a nível de LCU, aplica maior profundidade, às LCUs em regiões em que a atenção humana é maior, priorizando a qualidade subjetiva

do vídeo. Posteriormente, a solução foi aperfeiçoada com a implementação de um *feedback* a nível de quadro (DENG; XU; LI, 2016). Já um sistema de controle com base no ajuste dinâmico da profundidade CTU é observado em Zhang et al. (2018). A complexidade da CTU é estimada através de um modelo que considera o consumo de bits da PU 2Nx2N na profundidade 0. Um subconjunto de modos para predição inter-quadros é selecionado para atingir a complexidade alvo (ZHANG et al., 2018a). Um sistema semelhante de alocação e *feedback*, utilizando o SATD para estimar o tempo de codificação da CTU, foi apresentado em Zhang et al. (2019), porém com foco na predição intra-quadro.

Soluções de controle com base em algoritmos rápidos para divisão de CU também são explorados. Em Fang et al. (2018) é apresentada uma solução heurística, na qual a complexidade alvo é estimada a partir da média dos quatro primeiros quadros P e a complexidade é alocada a nível de quadro. Em Huang et al. (2019) é mostrado um modelo de decisão CU baseado em *Random Forest*, obtido a partir de um processo de aprendizagem online. O método utiliza o primeiro quadro de cada segmento codificado com o codificador original para atualizar os parâmetros de alocação de complexidade. Posteriormente, os autores obtêm novos resultados usando um conjunto mais amplo de sequências de teste Huang et al. (2021). Já outro sistema de controle, apresentado por Huang et al. (2021a) utiliza dois algoritmos de aceleração, um baseado em rede neural e outro em Naive Bayes, aplicados, respectivamente a CU e a PU.

Inteligência artificial também é usada em Li et al. (2020) para ajustar a complexidade de codificação em tempo real. Os autores propõem um algoritmo que promove a aceleração um modelo de última geração de redes neurais profundas voltado para o particionamento das CTUs.

A Tabela 7 apresenta informações sobre os trabalhos relacionados à controle de complexidade mencionados acima. As primeiras duas linhas da Tabela 7 mostram dados refentes aos trabalhos desenvolvidos para o VVC. Nas demais linhas são apresentados informações relacionadas aos artigos voltados para o HEVC, respeitando a ordem temporal de publicação. Nela podem ser observadas a etapa de codificação explorada para ajustar a complexidade, o nível de alocação de *budget* empregado, o software de referência utilizado para obter os resultados, assim como as faixas de RTC, erro médio e BD-Rate para cada artigo publicado.

Para ajustar a complexidade, quatro métodos distintos podem ser observados dentre os trabalhos da literatura: a decisão do tipo de particionamento, como visto em (HUANG et al., 2019), (FANG et al., 2018) e (ZHANG et al., 2018), a profundidade da árvore de particionamento como, por exemplo, (HUANG et al., 2022), (CHEN et al., 2018a) e (DENG et al., 2016), os modos intra-quadro (HOSSEINI et al., 2020) e os parâmetros de codificação utilizados, como em (GRELLERT et al., 2013), (GRELLERT et al., 2017) e (CORRÊA et al., 2014).

Algumas soluções não utilizam alocação de *budget* de tempo, como é caso de (CORRÊA et al., 2011), (CORRÊA et al., 2012) e (JIMENEZ-MORENO; MARTÍNEZ-ENRÍQUEZ; MARÍA, 2016). Outros trabalhos usam em único nível (CU, CTU, LCU, quadro, GOP ou segmento) como (GRELLERT et al., 2017), (DENG; XU; LI, 2016) e (ZHANG et al., 2018). Também há soluções que empregam alocação hierárquica em diferentes níveis como, por exemplo, (CHEN et al., 2018a), (HUANG et al., 2019) e (HUANG et al., 2021).

Os resultados são apresentados para diferentes faixas de redução de complexidade, a maior delas é de 10% a 90%, observada em (CORRÊA et al., 2014) e (CORRÊA et al., 2016). A precisão dos controladores também é bastante variável em relação ao alvo de redução, com valores de erro médio de 0 a 12,22 pontos percentuais. Já o BD-Rate varia de 0,07% a 18,11%.

Como mencionado anteriormente, não foi possível encontrar na literatura nenhum trabalho propondo um controlador adaptativo de complexidade para o codificador AV1. No entanto, a solução desenvolvida nesta tese explora diversas técnicas vistas nos artigos da literatura, entre elas:

- Utilização de um fluxo de controle semelhante ao implementado nas soluções de controle de complexidade e energia apresentadas, respectivamente, por Corrêa et al. (2016) e Penny et al. (2016).

- Alteração dos parâmetros de codificação para promover a redução de complexidade em diferentes faixas de atuação assim como (GRELLERT et al., 2013), (GRELLERT et al., 2017) e (CORRÊA et al., 2014).

- Utilização de frente de Pareto, também aplicado em (CORRÊA et al., 2016), (HOSSEINI et al., 2020) e (PENNY et al., 2016), para selecionar os pontos de operação com melhor relação entre de redução de tempo e eficiência de codificação.

- Alocação de *budget* de tempo a nível de GFG, similar a (CHEN et al., 2018a), (HUANG et al., 2019), (HUANG et al., 2021), os quais empregam alocação a nível de GOP.

Porém, é importante destacar que para viabilizar o desenvolvimento do controlador de complexidade AV1, as técnicas mencionadas acima necessitam ser adaptadas ao respectivo controlador. Além disso, o controlador apresentado nesta tese conta com outras inovações, como a predição do tempo de codificação e a especialização do controle de acordo com o vídeo de entrada, as quais serão detalhadas no Capítulo 6.

Tabela 7 – Resumo dos trabalhos encontrados na literatura referentes à controle de complexidade.

Artigo	**Etapa de Codif.**	**Alocação de *budget***	**Software de Referência**	**Faixa de RTC (%)**	**Faixa de Erro médio (± p.p.)**	**Faixa de BD-Rate (%)**
(HUANG et al., 2022)	Part. Intra	Quadro e CTU	VTM10.0	10 - 70	0,04 - 12,22	0,23 - 2,71
(HUANG et al., 2022a)	Part. Inter	GOP, quadro e LCU	VTM10.0	20 - 60	0,02 - 0,24	1,52 - 7,39
(CORRÊA et al., 2011)	Part.	–	HM 2.0	20 - 60	2,33 - 5,67	–
(CORRÊA et al., 2012)	Part.	–	HM 4.0	10 - 40	0 - 2	–
(CORRÊA et al., 2013)	Part.	–	HM 8.2	10 - 40	–	–
(GRELLERT et al., 2013)	Parâmetros de Codificação	CTU	HM 8.2	–	–	–
(CORRÊA et al., 2014)	Part.	–	HM 9.2	10 - 90	1,2 - 14	–
(CORRÊA et al., 2015)	Part.	–	HM 13	20 - 80	4,5 - 17,9	–
(CORRÊA et al., 2016)	Part.	–	HM 13	10 - 90	0,59 - 1,27	0,16 - 9,84
(JIMENEZ-MORENO; MARTÍNEZ-ENRÍQUEZ; MARÍA, 2016)	Part.	CU	HM 13.0	10 - 40	0,21 - 2,23	0,23 - 6,83
(DENG et al., 2016)	Part.	LCU	HM 14.0	20 - 80	0,73 - 2,54	0,84 - 17,49
(DENG; XU; LI, 2016)	Part.	Quadro	HM 16.0	20 - 80	0,54 - 1,49	0,53 - 10,24
(GRELLERT et al., 2017)	Parâmetros de Codificação	CTU	HM 10	20 - 60	0,38 - 1,52	0,65 - 7,13
(ZHANG et al., 2018)	Part.	CTU	HM 16.9	20 - 60	0,28 - 1,57	–
(ZHANG et al., 2018a)	Part.	CTU	HM 16.9	20 - 80	0,01 - 1,18	1,36 - 18,11
(FANG et al., 2018)	Part.	Quadro	HM 12.1	20 - 40	–	–
(CHEN et al., 2018a)	Part.	GOP, quadro, CTU	HM 13.0	10 - 50	0,17 - 1,22	0,50 - 3,48
(ZHANG et al., 2019)	Part.	CTU	HM 16.9	20 - 80	0,13 - 0,52	0,42 - 14,82
(HUANG et al., 2019)	Part. Intra	Segmento, GOP, quadro e CTU	HM 16	20 - 60	0,36 - 0,71	0,36 - 4,25
(HOSSEINI et al., 2020)	Modos Intra	Quadro	HM 16.14	20 - 40	0,01 - 0,03	0,28 - 3,69
(LI et al., 2020)	Part.	Quadro	HM 16.5	50 - 95	0,23 - 0,81	2,12 - 3,1
(PAKDAMAN et al., 2021)	Part. Intra	Quadro	HM 16.14	20 - 80	0 - 0,23	0,08 - 11,71
(HUANG et al., 2021)	Part.	Segmento, GOP, quadro e CTU	HM 16.0	20 - 60	0,52 - 1,01	0,41 - 5,04
(HUANG et al., 2021a)	Part. Intra	CU	HM 16.5	40 - 60	1,7 - 5,6	0,07 - 1,9

5 ANÁLISE DE COMPLEXIDADE DO CODIFICADOR AV1

Investigar e entender a distribuição da complexidade do software de referência Liboam auxilia a definir, com mais precisão, onde aplicar os esforços de otimização no codificador AV1. A identificação dos estágios de codificação que possuem maior complexidade e, por consequência, consomem uma porção considerável do tempo de codificação, pode sinalizar onde soluções de redução e controle de complexidade devem focar para atingir melhores resultados. Como o AV1 é um codificador ainda recente, não foram encontrados na literatura trabalhos relacionados à sua complexidade quando esta pesquisa iniciou.

Diante disso, foi necessário realizar uma análise de complexidade a partir do *profiling* das etapas de codificação, visando compreender os esforços computacionais empregados nos diferentes estágios do processo de codificação do AV1. A seguir, foi avaliada a distribuição na estrutura de particionamento do AV1. Por fim, foram avaliados os custos computacionais e a eficiência de codificação relacionadas a diferentes parâmetros de codificação utilizados no AV1.

5.1 Setup de Experimentos e Metodologia

Esta seção relata a metodologia usada nas análises de complexidade realizadas neste trabalho. As avaliações foram feitas utilizando os primeiros sessenta quadros de dez sequências de vídeo com resoluções HD 1080 (1920×1080 pixels) e UHD 4K (4096×2160 pixels). Todas as sequências usadas são recomendadas pelo IETF-NETVC-Testing (DAEDE; NORKIN; BRAILOVSKIY, 2020) para experimentos com o AV1. Para garantir que as sequências de vídeo diferem entre si nas características espaciais e temporais, uma análise do índice de atividade espacial (SI - *Spatial Index*) e do índice de atividade temporal (TI - *Temporal Index*) (WEBSTER; WOLF, 1992) foi realizada. A Figura 14 ilustra essa relação (com o SI no eixo X e o TI no eixo Y) resultante das 21 sequências de vídeo HD 1080 (Figura 14a) e das oito sequências de vídeos UHD 4K (Figura 14b) definidas pelo IETFNETVC-Testing. As sequências de vídeo utilizadas nas análises estão circuladas em vermelho.

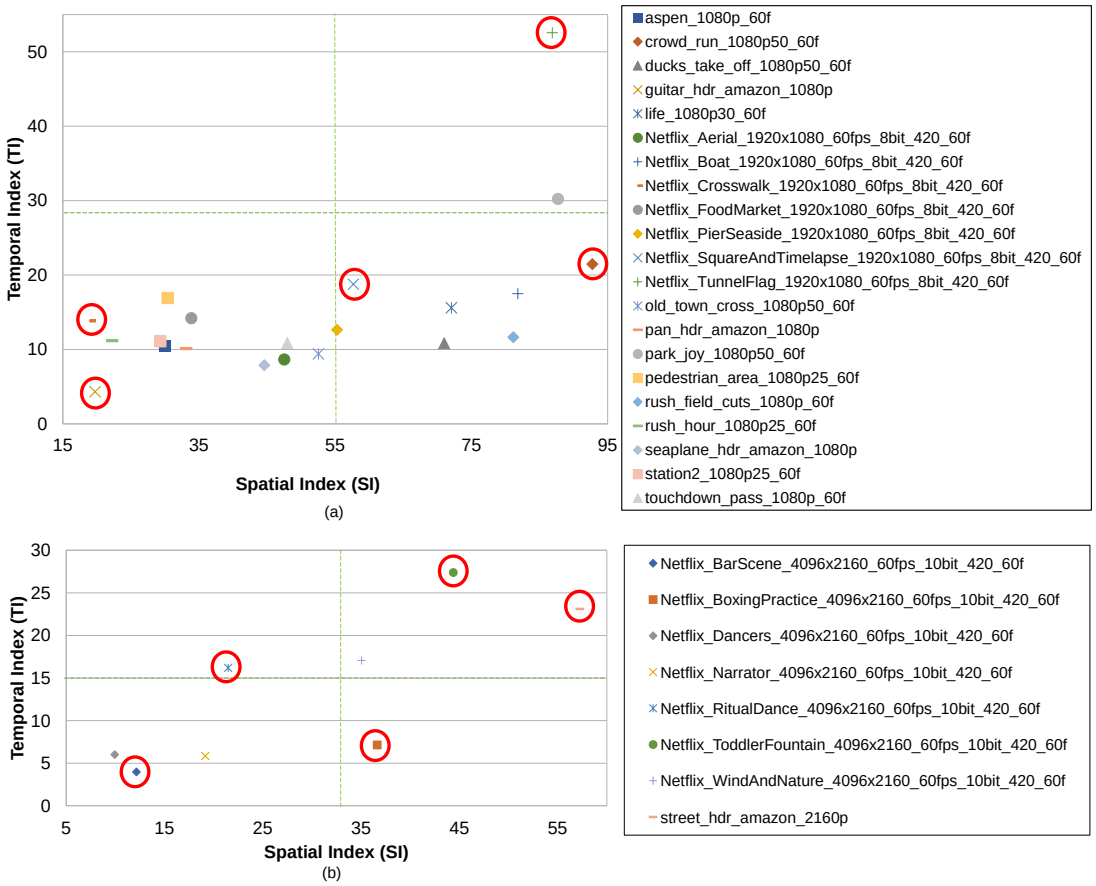


Figura 14 – Índice de atividade espacial e temporal para sequências de vídeos recomendadas em (DAEDE; NORKIN; BRAILOVSKIY, 2020) (a) Resolução HD 1080 (b) Resolução UHD 4K.

A análise *profiling* (apresentada na Seção 5.2), foi realizada usando quatro sequências de cada resolução. Para HD 1080, foram usadas as sequências: *Netflix_TunnelFlag, Guitar_Hdr_Amazon, Crowd_Run* e *Netflix_Crosswalk*. Para UHD 4K, as sequências de vídeo selecionadas foram: *Netflix_ToddlerFountain, Netflix_BarScene, street_hdr_amazon* e *Netflix_RitualDance*. Conforme pode ser observado na Figura 14, estas sequências apresentam perfis de informação temporal e espacial bastante diferentes, garantindo um espectro de vídeos suficientemente distintos para suportar as conclusões das avaliações realizadas. As demais análises apresentadas neste trabalho, foram realizadas considerando cinco sequências de vídeo para cada resolução. As seguintes sequências de vídeos foram usadas para a resolução HD 1080: *Netflix_TunnelFlag, Guitar_Hdr_Amazon, Crowd_Run, Netflix_Crosswalk e Netflix_SaquereAndTimelapse*. Para a resolução UHD 4K foram usados os vídeos: *Netflix_ToddlerFountain, Netflix_BarScene, street_hdr_amazon, Netflix_RitualDance* e *Netflix_BoxingPractice*.

A expansão do conjunto de sequências de vídeo foi realizada com o objetivo de aperfeiçoar a distribuição do perfil das sequências analisadas, onde as novas sequências incluídas privilegiaram perfis de SI e de TI medianos, que não estavam presentes nos conjuntos originais. Para o desenvolvimento das avaliações do AV1 apresentadas

neste trabalho, foram realizadas 2384 codificações. Todas as avaliações foram realizadas no Libaom com o perfil *High Latency* CQP e com valores de CQ 20, 32, 43 e 55, conforme recomendado no IETF-NETVC-Testing (DAEDE; NORKIN; BRAILOVSKIY, 2020) e no artigo de referência do AV1 (CHEN et al., 2020).

5.2 Análise do Tempo de Execução dos Estágios de Codificação AV1

Para analisar o tempo exigido por cada estágio (intra, inter, transformadas, quantização, filtros e entropia) foi utilizada a ferramenta Gprof (GPROF, 2009), que suporta a mesma linguagem de programação usada no Libaom (linguagem C) e já foi aplicada, especificamente, em codificação de vídeo, como pode ser observado em (GRELLERT et al., 2013) e (AYADI et al., 2018).

O Gprof é uma ferramenta voltada à análise dinâmica de execução de programas (*profiler*) e reporta informação de tempo de execução de cada uma das funções do programa. Para cada arquivo extraído do Gprof, usando a ferramenta Gprof2dot (FONSECA, 2020), foi possível gerar um arquivo gráfico contendo a árvore de codificação, como apresentado na Figura 15. Nessas árvores são apresentadas, de cima para baixo, as rotinas em ordem de chamada. Cabe destacar que no Gprof é possível navegar entre as rotinas e aplicar zoom, viabilizando, assim, observar as funções utilizadas durante o processo de codificação e verificar o tempo de execução de uma dada função e de suas sub-rotinas, como exemplifica a Figura 15. Devido à baixa resolução da imagem completa apresentada na Figura 15, um destaque de um pequeno trecho também foi apresentado, no qual pode ser observado informações referentes as funções *av1_rd_pick_inter_mode_sb* e *Handle_inter_mode*.

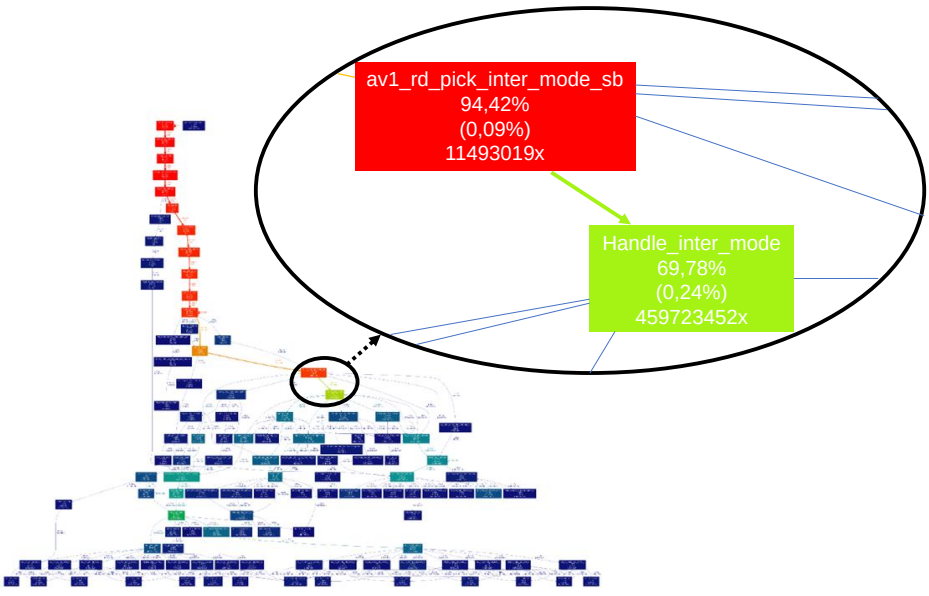


Figura 15 – Exemplo de gráfico gerado após análise Gprof.

Para a construção da árvore de codificação, o Gprof considera apenas as funções de maior importância. Ainda assim, o grande número de funções existentes no Libaom (cerca de 1200 funções) e a coexistência delas em diferentes etapas de codificação, dificulta classificá-las nas diversas etapas presentes no fluxo de codificação do AV1 (Figura 8). No caso da classificação manual, a dificuldade está relacionada ao grande número de funções. Já a classificação automatizada é prejudicada pela falta de padronização nas nomenclaturas das funções e a ausência de identificação clara das etapas de codificação no Libaom. Portanto, os resultados apresentados a seguir, obtidos através da análise Gprof, são aproximados.

A fim de viabilizar a análise dos tempos de codificação de cada estágio, uma árvore de codificação resumida foi construída com base em vinte "funções chaves". Essas funções, facilmente catalogadas por estágio de codificação, foram identificadas e selecionadas a partir de uma análise exaustiva de todos os resultados. A Tabela 8 apresenta as funções selecionadas e os respectivos estágios de codificação a que pertencem.

Tabela 8 – Distribuição das funções chaves Gprof nos estágios de codificação realizada manualmente.

	Nome da Função
Intra	av1_rd_pick_intra_mode_sb rd_pick_intra_sbuv_mode rd_pick_intra_angle_sby super_block_yrd super_block_uvrd av1_encode_intra_block_plane
Inter	av1_rd_pick_inter_mode_sb
Transformadas	av1_foreach_transformed_block_in_plane search_txk_type av1_inverse_transform_block av1_xform_quant
Quantização	av1_quantize_fp_facade av1_quantize_b_facade av1_highbd_quantize_fp_facade av1_highbd_quantize_b_facade
Entropia	av1_cost_coeff_txb aom_codec_av1_cx
Filtros	av1_temporal_filter av1_cdef_search av1_pick_filter_restoration

As Tabelas 9 e 10 apresentam os resultados, respectivamente, para resolução HD 1080 e UHD 4K, obtidos a partir de um servidor de alto poder de processamento que conta com um processador Intel Xeon E5645, com seis núcleos de 2,4 GHz e 24 GB de RAM. Nelas, é possível observar os resultados médios do tempo percentual de codificação (%) e do tempo absoluto de codificação (abs) dos quatro CQs analisados, para cada estágio de codificação de vídeo AV1. Os valores absolutos estão expressos em horas.

Tabela 9 – Resultados médios da análise Gprof para as sequências de teste na resolução HD 1080.

	CQ 20		CQ 32		CQ 43		CQ 55	
	%	abs	%	abs	%	abs	%	abs
Intra	0,37	0,13	0,52	0,15	0,83	0,16	1,28	0,18
Inter	71,59	25,2	77,42	22,46	79,52	15,42	79,42	11,19
Transf.	24,97	8,79	19,79	5,74	17,28	3,35	16,11	2,27
Quant.	1,79	0,63	1,10	0,32	0,67	0,13	0,57	0,08
Filtros	1,25	0,44	1,14	0,33	1,70	0,33	2,62	0,37
Entropia	0,03	0,01	0,03	0,01	0,00	0,00	0,00	0,00
TOTAL	100,00	35,2	100,00	29,01	100,00	19,39	100,00	14,09

Tabela 10 – Resultados médios da análise Gprof para as sequências de teste na resolução UHD 4K.

	CQ 20		CQ 32		CQ 43		CQ 55	
	%	abs	%	abs	%	abs	%	abs
Intra	2,69	6,13	1,28	1,77	0,68	0,57	2,22	1,44
Inter	51,10	116,51	62,06	85,58	71,79	60,16	72,81	47,18
Transf.	43,60	99,41	34,12	47,05	24,64	20,65	20,52	13,30
Quant.	1,69	3,85	1,72	2,37	1,20	1,01	0,84	0,54
Filtros	0,87	1,98	0,79	1,09	1,68	1,41	3,60	2,33
Entropia	0,05	0,11	0,03	0,04	0,01	0,01	0,01	0,01
TOTAL	100,00	227,99	100,00	137,9	100,00	83,81	100,00	64,8

Juntos, a predição intra-quadro, a quantização, os filtros e a codificação de entropia utilizam 3,48% e 4,48% do tempo total de execução do Liboam para as resoluções HD 1080 e UHD 4K, respectivamente. A predição intra-quadro corresponde a apenas 0,75% do tempo total de codificação para as resoluções HD 1080 e, na resolução UHD 4K, este valor é de 1,72%. O pequeno percentual referente à predição intra-quadro era esperado, pois este tipo de predição precisa processar apenas informações dentro de um único quadro, o que é menos complexo do que analisar informações em vários quadros diferentes como, por exemplo, faz a predição inter-quadros. Além disso, o percentual de tempo da predição intra-quadros é impactado por algoritmos de decisão rápida presentes no Libaom. Segundo Grellert et al. (2013), a predição intra-quadros do HEVC corresponde a 3% do tempo total de execução. No VVC o tempo de codificação desta etapa é 108 vezes maior que no HEVC (SIQUEIRA I.AND CORRÊA; GRELLERT, 2020).

Já o estágio de quantização é um processo simples, que atenua os coeficientes transformados de acordo com o valor CQ. Além disso, o Libaom pré-calcula vários parâmetros necessários, agilizando o processamento. Portanto, é compreensível que este estágio de codificação represente cerca de 1% do tempo de execução Libaom para ambas as resoluções.

Os filtros de laço requerem 1,68% e 2,57% do tempo de processamento Libaom para as resoluções HD 1080 e UHD 4K, respectivamente. Em geral, à medida que

o CQ aumenta, é possível observar um aumento do tempo dedicado pelo Libaom a este estágio de codificação. A razão para este crescimento é que quanto maior o CQ, maiores são os artefatos gerados pela quantização e, assim, as o esforço para atenuar esses artefatos tornam-se cada vez mais necessárias.

A codificação de entropia é responsável por tratar todos os dados processados pelo codificador, entretanto, sem a necessidade de testar diferentes modos de operação. Este fato justifica o pequeno tempo de codificação consumido por este estágio em ambas as resoluções, com cerca de 0,2% do tempo total.

Diante do que foi exposto no Capítulo 3, o alto custo computacional da predição inter-quadros já era esperado, e isso pode ser comprovado nos resultados apresentados. De acordo com os resultados na Tabela 9 e na Tabela 10, o custo computacional relativo da predição inter é diretamente influenciado pelo nível de CQ utilizado e pela resolução do vídeo. Quanto maior é o valor de CQ, o custo computacional relativo da predição inter também tende a ser maior. Por outro lado, quanto maior a resolução, menor tende a ser o custo relativo da predição inter. Em ambas as resoluções de vídeo, e nos quatro níveis de CQ avaliados, no mínimo 51,1% do tempo de codificação é dedicado à predição inter-quadros. Este valor pode chegar a 79,52%, no caso máximo observado nos experimentos. Este estágio de codificação requer 76,99% e 64,44% do tempo médio de execução para codificar vídeo HD 1080 e UHD 4K, respectivamente, quando considerados os quatro CQs.

O custo computacional do estágio de transformadas foi a principal surpresa nos resultados experimentais. Em função do elevado número de combinações de tipos de transformadas e de tamanhos de bloco, as transformadas ocupam, com folga, o segundo lugar na complexidade do AV1. Quanto maior o valor de CQ, menor é o custo computacional relativo das transformadas. O custo referente a esta etapa de codificação varia entre 16,11% e 43,6% do tempo total de execução do Libaom. Os tempos médios de execução para codificar vídeos HD 1080 e UHD 4K são, respectivamente, 19,54% e 30,72%.

A Figura 16 apresenta uma análise da alocação do tempo de codificação das funções *convolve*, *compound* e subpixel, utilizadas na predição inter-quadros, e das funções DCT inversa e DCT direta, usadas nas transformadas. Essas funções são as que apresentam os maiores tempos de execução nas suas respectivas etapas de codificação. Os resultados médios foram obtidos a partir dos vídeos HD 1080 e CQs mencionados na Seção 5.1 deste trabalho. Para o CQ 20, a função *convolve*, apresenta o tempo médio de codificação de 23,60%, a função *compound* 21,31% e a função subpixel 4,91%. Esses valores são similares aos demais CQs, como pode ser visto na Figura 16.

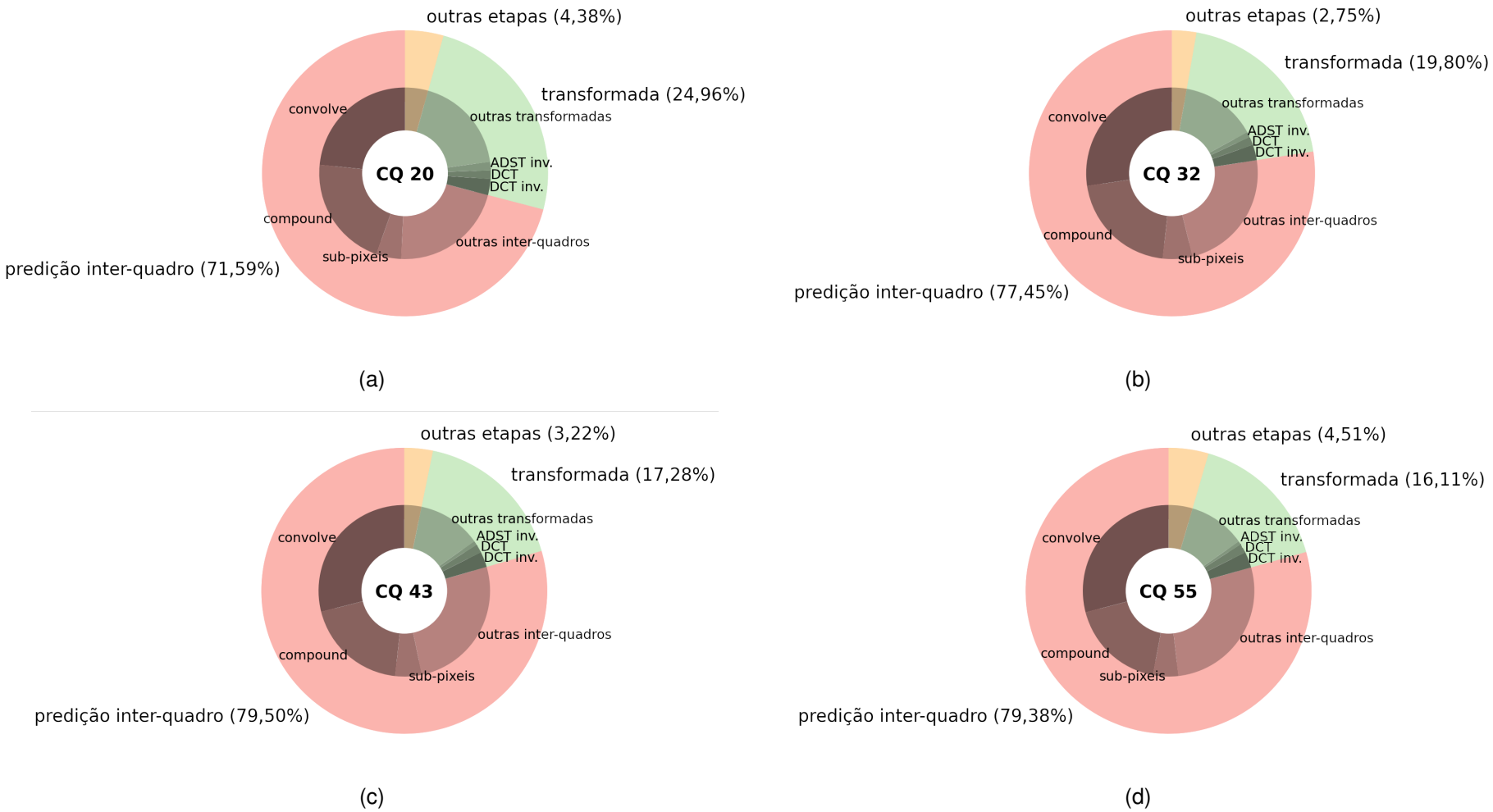


Figura 16 – Distribuição da porcentagem do tempo de codificação das funções utilizadas na etapa de predição inter-quadros e transformadas para resolução HD 1080 (a) CQ 20, (b) CQ 32, (c) CQ 43 e (d) CQ 55.

A função *convolve* está relacionada com a operação matemática conhecida como convolução (SMITH, 1997) (OPPENHEIM; SCHAFER; STOCKHAM, 1968), que aplica uma interpolação subamostrada de duas convoluções unidirecionais. Primeiro, um filtro horizontal é usado para construir um *array* temporário e, então, este *array* é verticalmente filtrado para obter a predição final (RIVAZ; HAUGHTON, 2018). A função *compound* está relacionada ao modo composto do AV1, o qual possibilita combinar quadros de referência. Um preditor para frente e para trás são considerados para a combinação (LIN et al., 2018). Já a função subpixel é responsável pela busca em pixels fracionários do AV1.

Na etapa de transformadas, a DCT inversa e a DCT direta são as funções mais complexas, representando em média, 3,08% e 1,62% do tempo total de codificação para o CQ 20, respectivamente. O fato da DCT ser o tipo de transformada que requer mais tempo de codificação é esperado, pois em Zhao e Liu (2020), onde é realizada uma análise específica das transformadas em blocos intra-quadro, a DCT tipo 2 (DCT-II) está entre as transformadas primárias mais selecionadas no AV1.

Uma observação global importante é que, independentemente da resolução considerada, quanto maior o valor de CQ, menor é o custo computacional total da codificação. Isso ocorre, principalmente, pelo fato que CQs maiores tendem a eliminar mais informações, gerando áreas mais homogêneas nos vídeos, facilitando o processo de codificação. No entanto, é importante destacar que mesmo usando o maior CQ, o tempo de codificação requerido é bastante significativo, principalmente para as sequências de vídeo com resolução UHD 4K. Usando o Libaom, *hash* cd653f1 e sequências de teste compostas por 60 quadros, o tempo médio de codificação para a resolução UHD 4K obtido a partir do maior CQ (CQ=55) é de, aproximadamente, três dias. Caso seja utilizado o menor CQ (CQ=20) o período é estendido para 10 dias, aproximadamente.

Outra observação importante, com base nos resultados apresentados nesta seção, é que a complexidade do Libaom está distribuída de maneira bastante heterogênea entre os estágios de codificação. Assim, é possível perceber que otimizações voltadas à redução de complexidade para os estágios de predição intra-quadro, quantização, filtros e codificação de entropia tendem a ter ganhos globais pequenos. Em contrapartida, soluções voltadas à redução de complexidade dos estágios de codificação inter-quadros e transformadas podem apresentar resultados mais expressivos.

5.3 Estrutura de Particionamento

Diante dos resultados apresentados na Seção 5.2 e do fato que tanto a predição inter-quadros quanto a etapa de transformadas são aplicadas em blocos com diferentes tamanhos e formatos, foi considerado pertinente observar como as decisões de

particionamento ocorrem no AV1. Os parâmetros de particionamento, assim como os parâmetros específicos às etapas de predição inter-quadros e transformadas foram explorados para determinar os Pontos de Controle das soluções apresentadas neste trabalho, logo, é importante analisá-los.

Para tanto, foram extraídas do Libaom as informações relativas ao tamanho do bloco e o tipo de particionamento escolhidos em ambos estágios de codificação citados, com exceção do modo SPLIT, o qual não é armazenado explicitamente pelo codificador. Posteriormente, foi contabilizado quantas vezes cada uma das possibilidades de escolha de particionamento de bloco foram selecionadas como melhor opção. A representatividade de cada particionamento foi calculada a partir do número de ocorrências de cada particionamento multiplicado por um fator de área correspondente ao respectivo tamanho de bloco. Para blocos 4×4 (menor bloco suportado) foi atribuído um fator 1, enquanto para blocos 128×128, por exemplo, foi atribuído o fator 1024, dado que este ocupa uma área 1024 vezes maior que o bloco 4×4.

A Figura 17 apresenta o percentual da área do vídeo codificada com diferentes tipos de particionamento do AV1 nos estágios de predição inter-quadros e transformadas, considerando os quatro CQs analisados. No eixo X, são apresentados os particionamentos possíveis, conforme apresentado no Capítulo 3, e no eixo Y é apresentado o percentual de área dos *frames* dos vídeos que são codificadas usando cada tipo de particionamento. A primeira conclusão com a análise da Figura 17 é que, em ambas as resoluções avaliadas, cada tipo de particionamento é utilizado de maneira semelhante. Na grande maioria das vezes, em média 60,67%, o codificador mantém a razão 1:1 (NONE), ou seja, usa blocos quadráticos. O segundo tipo de partição mais usado é o vertical (VERT), com média de 14,74%, seguido pela partição horizontal (HORZ), com média de 11,48%. Os blocos horizontais 1:4/4:1 são os menos aplicados e correspondem, em média, a 0,69% do total dos blocos usados na predição inter-quadros e transformadas.

A Figura 18 apresenta o percentual da área do vídeo codificada com os tamanhos de blocos permitidos na predição inter-quadros do AV1. Nesse caso, o eixo X apresenta todos os tamanhos de blocos possíveis, quando aplicado o particionamento para a predição inter-quadros. O eixo Y, novamente apresenta o percentual de área dos quadros que é codificado usando cada tamanho de bloco. Para predição inter-quadros das sequências de vídeos com resolução UHD 4K, o uso dos tamanhos de blocos 4×4, 4×8, 8×4, 4×16 e 16×4 é muito pequeno, a ponto de não ser visto no gráfico. Além disso, os blocos 64×128 e 128×64 são usados para codificar uma pequena parcela da área dos vídeos, com 0,03% para ambos. O tamanho de bloco usado para codificar a maior área dos vídeos, quando da predição inter-quadros, é o 128×128, correspondendo a 46,06% da área codificada. Já os blocos 64×64 são usados para codificar apenas 13,95% da área. Este cenário é diferente para vídeos com resolução

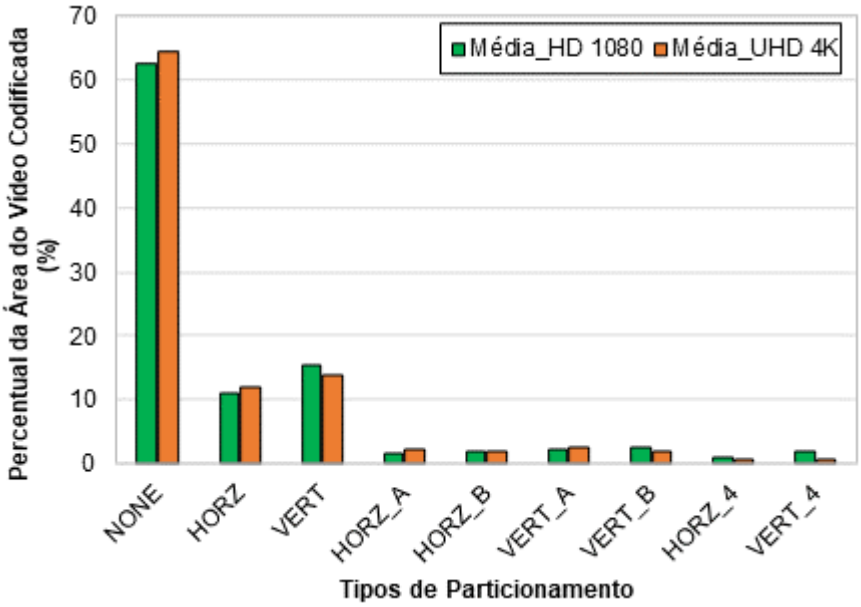


Figura 17 – Percentual da área do vídeo codificada com diferentes tipos de particionamento do AV1 nos estágios de predição inter-quadros e transformadas

HD 1080. Nesta resolução, as áreas codificadas por blocos 128×128 e 64×64 são similares (ou seja, os blocos 64×64 são muito mais usados que os 128×128, uma vez que são quatro vezes menores). O primeiro é aplicado para codificar 20,35% da área dos vídeos e o segundo é usado para codificar 21,68% da área. Outra diferença é que nas sequências de vídeos com resolução HD 1080, todos os tamanhos de blocos são utilizados de forma representativa para codificar os vídeos, ou seja, podem ser observados no gráfico.

Também é possível observar, a partir da Figura 18, que quanto maior é a resolução do vídeo, maior é a tendência de uso de blocos maiores para codificar os vídeos, o que é esperado, uma vez que vídeos com resoluções maiores tendem a ter mais áreas homogêneas que tendem a ser codificadas com tamanhos de blocos maiores.

A Figura 19 apresenta o percentual da área do vídeo codificada com os tamanhos de blocos utilizados pelo estágio de transformadas. Novamente, o eixo X apresenta todos os tamanhos de blocos possíveis durante a etapa de transformada e o eixo Y apresenta o percentual da área do vídeo codificada usando cada tamanho de bloco. Como pode ser observado na Figura 18, os blocos menores são mais explorados na etapa de transformadas. Considerando a resolução HD 1080, os blocos 4×4, por exemplo, correspondem a 0,17% do total da área codificada na predição inter-quadros. Já nas transformadas, esse tamanho de bloco é usado em 1,23% da área. Na resolução UHD 4K, este valor passa a ser 2,63%. Na etapa de transformadas, os blocos 64×64 são os utilizados para codificar a maior área em ambas as resoluções, com 22,82% e 24,33%, para a resolução HD 1080 e UHD 4K, respectivamente. Em ambas as resoluções, os tamanhos de bloco 8×8, 16×16, 32×32 e 64×64 são os mais repre-

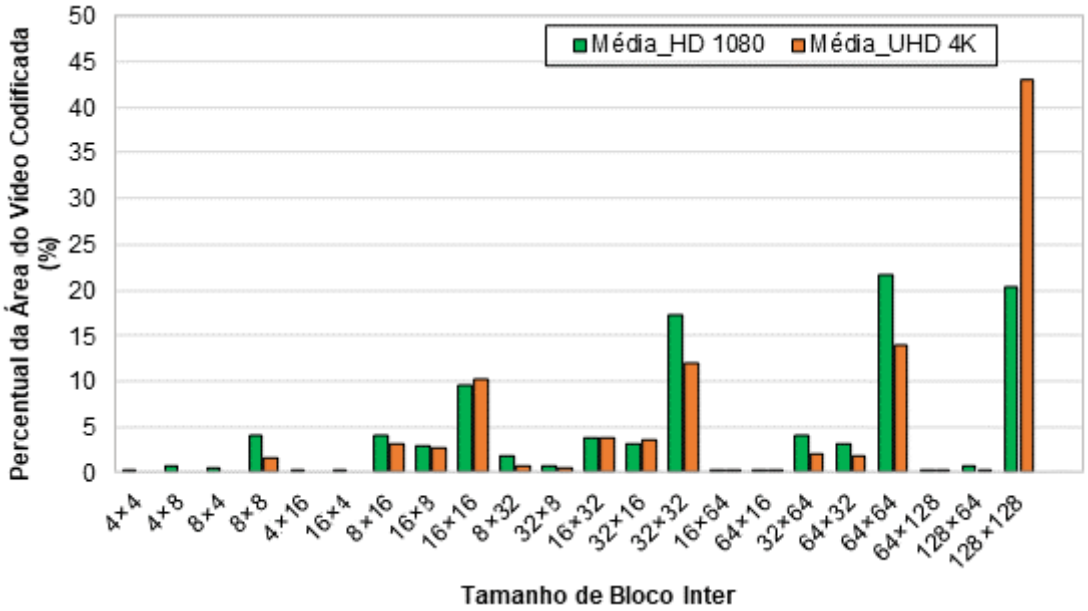


Figura 18 – Percentual da área do vídeo codificada para cada tamanho de bloco permitido na predição inter-quadros do AV1.

sentativos da codificação, embora com índices de uso diferentes em cada caso. Uma curiosidade é a expressiva utilização de blocos 16×16 na resolução UHD 4K.

Para melhor investigar esse comportamento das transformadas, a Figura 20 apresenta os mesmos resultados da Figura 19, detalhando os resultados para cada sequência de vídeo. Na Figura 20 é possível perceber que o comportamento das transformadas é completamente dependente das características do vídeo, com uma variação expressiva nas áreas codificadas com cada tamanho de bloco. Por exemplo, a elevada representatividade dos blocos 16×16, na resolução UHD 4K, conforme apresentado na Figura 19, é explicado na Figura 20 por conta da grande influência dos vídeos *Netflix_RitualDance* e *Netflix_BoxingPractice* que impactam nos resultados médios apresentados na Figura 19. Além de apresentarem índices espaciais e temporais próximos, ambos vídeos usam movimento de câmera, o que pode resultar na necessidade de particionamentos menores. Blocos 8×8 também são mais representativos nesses dois vídeos do que nos demais.

Os resultados dessas análises iniciais apontam para importantes oportunidades para desenvolver soluções para controle adaptativo da complexidade de codificadores AV1, uma vez que, dependendo do CQ, da resolução e das características dos vídeos, o codificador toma decisões distintas, com prováveis impactos no tempo de execução. Assim, uma avaliação mais profunda sobre o impacto das ferramentas e dos parâmetros de codificação sobre a complexidade e a eficiência de codificação são essenciais para melhor compreender as oportunidades relacionadas ao desenvolvimento de soluções para controle de complexidade. Essas análises estão presentes na próxima Seção 5.4 desta tese.

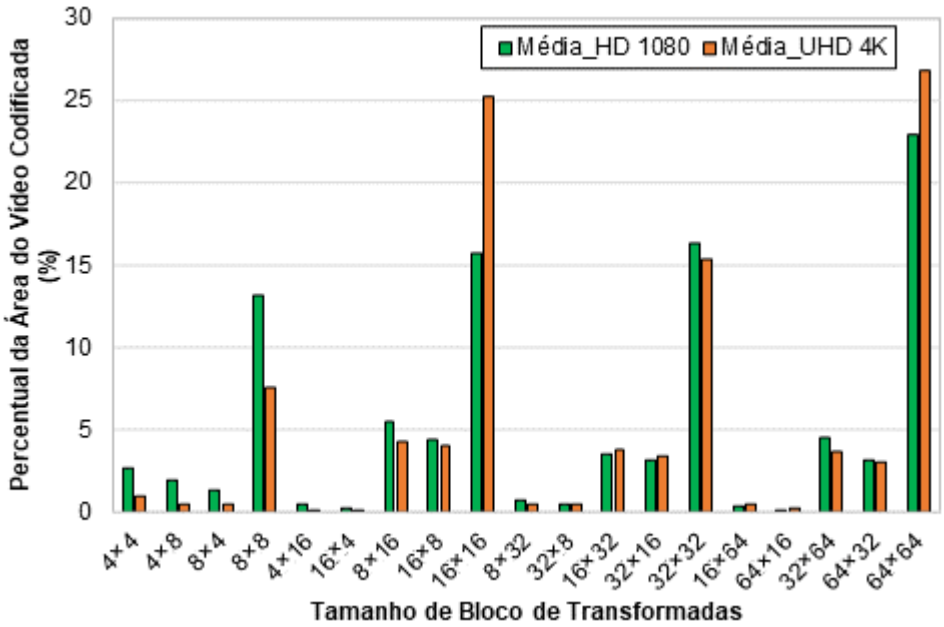


Figura 19 – Percentual da área do vídeo codificada para cada tamanho de bloco permitido no estágio de transformadas do AV1.

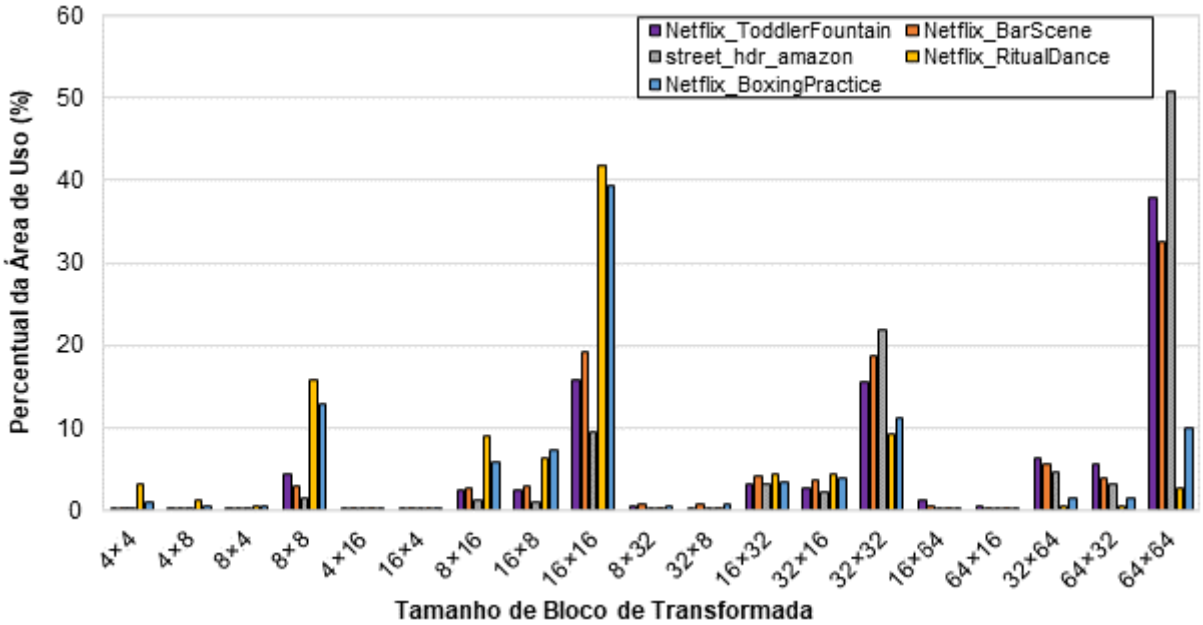


Figura 20 – Percentual da área do vídeo codificada para os tamanhos de blocos do estágio de transformadas do AV1 relativos às sequências de vídeo UHD 4K.

5.4 Sensibilidade do AV1 em relação aos Parâmetros de Codificação

Esta seção avalia o comportamento da complexidade do codificador AV1, em relação ao tempo de codificação, diante da variação dos principais parâmetros de codificação disponíveis no Libaom. Os resultados deste processo são importantes para o desenvolvimento do controlador, que é o principal objetivo deste trabalho, uma vez que o ajuste de complexidade é realizado a partir da seleção de Pontos de Controle compostos por diferentes parâmetros de codificação. Os parâmetros de codificação do Libaom, presentes no arquivo de configuração, estão detalhados no Apêndice A desta tese.

Inicialmente, a maioria dos parâmetros foram configurados individualmente como, por exemplo, o parâmetro *dist-wtd-comp*, o qual habilita/desabilita o modo de predição composta inter-quadros denominada *Distance Weighted Predictor* e o parâmetro *1to4-partitions*, o qual habilita/desabilita as partições do tipo 1:4 e 4:1. Apenas os parâmetros *min-partition-size* e *max-partition-size*, os quais definem, respectivamente, o tamanho mínimo e máximo de partição, foram modificados simultaneamente para limitar a profundidade da árvore de particionamento. As configurações foram geradas a partir da alteração do estado padrão de cada um dos parâmetros analisados. Os parâmetros *dist-wtd-comp* e *1to4-partitions*, por exemplo, por padrão, encontram-se habilitados no Libaom. Ao serem desabilitados geraram duas configurações distintas, denominadas *c19* e *c23*, respectivamente. Assim, foram criadas e analisadas 39 configurações, detalhadas no Apêndice B, partindo da configuração *c0*, que é a configuração padrão, ou seja, corresponde às condições de teste encontradas nos arquivos de configuração originais do Libaom, chegando até à configuração *c38*. Todos os parâmetros foram modificados, a cada nova configuração, com o objetivo de reduzir a complexidade do codificador, quando comparados à configuração padrão *c0*.

A Tabela 11 mostra, para cada um dos quatro CQs avaliados, o tempo de codificação normalizado em relação à configuração *c0*. Os resultados são apresentados tanto para resolução HD 1080 quanto para UHD 4K. Pode-se observar que o tempo de codificação difere entre os CQs e entre as resoluções. O tempo de codificação apresentado obtido para cada configuração, na maioria dos casos, foi menor que o tempo de codificação utilizado pela configuração *c0*, indicando que os parâmetros analisados promovem impacto na redução da complexidade do codificador. Algumas configurações reduzem o tempo de codificação de maneira bastante significativa. A configuração *c5*, por exemplo, para o CQ 20 da resolução HD 1080, diminuiu cerca de 78% o tempo de codificação. Este resultado era esperado, pois esta configuração limita o tamanho mínimo de bloco em 64×64, ou seja, o codificar está apto a trabalhar apenas com dois tamanhos de blocos: 64×64 (tamanho mínimo) e 128×128 (tamanho

máximo).

Porém, analisar apenas a redução de tempo não é suficiente para concluir se determinada configuração é adequada ou não para ser utilizada no processo de codificação. Também é importante observar a relação entre a taxa de bits (*bitrate*) e a qualidade do vídeo codificado, por isso, o uso de métricas de comparação como BD-Rate tornam mais fáceis a avaliação dos resultados. Nas Figuras 21 e 22 podem ser observados, para cada configuração proposta, os valores obtidos para a redução de tempo de codificação (RTC) e o BD-Rate, respectivamente. Estes valores estão descritos na Tabela 12, na qual também estão incluídos valores relativos à razão entre redução de tempo médio e o BD-Rate médio obtido. Cabe salientar que as configurações *c23* e *c28,* as quais alteram, respectivamente, os parâmetros *smooth-interintra* e *warped-motion*, apresentaram acréscimo no tempo médio de codificação. Por este motivo, essas configurações foram excluídas das análises seguintes apresentadas por este trabalho.

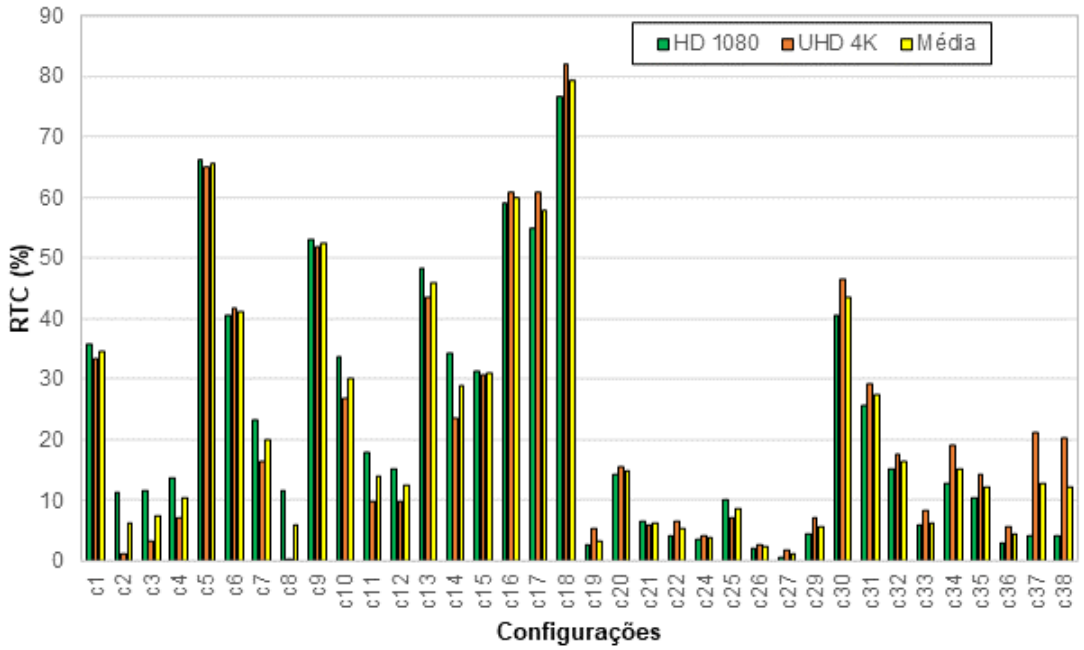


Figura 21 – Redução do tempo de codificação, em relação a configuração *c0*, obtido para as configurações propostas.

Nas configurações *c1* a *c18* são alterados parâmetros relacionados à estrutura de particionamento (Apêndice B). As configurações *c2* e *c3* promovem uma pequena redução no tempo médio total de codificação, quando comparadas às outras configurações, entre 6,18% e 7,29%, respectivamente. A configuração *c2* desativa partições AB, enquanto a configuração *c3* elimina o uso de partições 1:4/4:1. Já a configuração *c1* desativa os particionamentos retangulares e, consequentemente, impossibilita o uso tanto de partições do tipo AB quanto de partições 1:4/4:1. Como esperado, essa configuração é mais eficiente na redução do tempo de codificação, promovendo redu-

Tabela 11 – Tempo de codificação normalizado, média dos vídeos HD 1080 e UHD 4K, para cada um dos quatro CQs analisados.

Config.	Tempo normalizado - HD1080				Tempo normalizado - UHD4K			
	CQ 20	CQ 32	CQ 43	CQ 55	CQ 20	CQ 32	CQ 43	CQ 55
c1	0,51	0,60	0,72	0,87	0,57	0,60	0,77	0,79
c2	0,86	0,87	0,96	0,95	0,96	0,99	0,97	0,95
c3	0,84	0,86	0,96	0,97	0,96	0,93	0,97	1,00
c4	0,88	0,86	0,89	0,84	0,98	0,85	0,96	0,75
c5	0,22	0,28	0,46	0,51	0,25	0,32	0,48	0,61
c6	0,42	0,54	0,77	0,78	0,41	0,59	0,73	0,83
c7	0,60	0,82	0,89	0,81	0,73	0,83	0,90	0,95
c8	0,85	0,86	0,99	0,89	1,00	0,95	1,00	1,00
c9	0,36	0,49	0,57	0,52	0,39	0,48	0,55	0,59
c10	0,59	0,65	0,79	0,68	0,71	0,72	0,74	0,73
c11	0,84	0,82	0,89	0,76	0,99	0,85	0,83	0,77
c12	0,89	0,85	0,91	0,77	0,99	0,85	0,83	0,77
c13	0,51	0,47	0,56	0,57	0,72	0,46	0,50	0,49
c14	0,76	0,64	0,67	0,56	1,02	0,59	0,58	0,54
c15	0,81	0,67	0,69	0,57	0,87	0,59	0,58	0,54
c16	0,53	0,37	0,40	0,34	0,56	0,31	0,32	0,33
c17	0,58	0,40	0,42	0,43	0,56	0,31	0,32	0,33
c18	0,29	0,19	0,26	0,24	0,22	0,13	0,20	0,25
c19	1,04	0,94	0,96	0,96	0,96	0,96	0,99	0,97
c20	0,91	0,81	0,86	0,87	0,84	0,83	0,87	0,86
c21	0,98	0,93	0,94	0,91	0,95	0,94	0,93	0,92
c22	1,00	0,93	0,96	0,97	0,90	0,96	0,97	0,95
c23	1,07	0,99	1,01	1,00	0,98	1,00	1,02	0,99
c24	1,02	0,94	0,96	0,95	0,97	0,96	0,96	0,94
c25	0,94	0,85	0,91	0,93	0,93	0,92	0,95	0,95
c26	1,02	0,97	0,98	0,98	0,97	0,98	0,98	0,96
c27	1,05	0,98	0,98	0,98	0,99	0,98	0,98	0,97
c28	1,06	0,97	1,00	1,00	1,00	1,00	0,99	0,98
c29	1,00	0,92	0,96	0,96	0,92	0,95	0,94	0,93
c30	0,68	0,56	0,57	0,57	0,59	0,49	0,52	0,52
c31	0,84	0,72	0,73	0,73	0,76	0,67	0,69	0,69
c32	0,92	0,81	0,84	0,84	0,86	0,83	0,82	0,80
c33	1,00	0,91	0,93	0,93	0,94	0,92	0,98	0,90
c34	0,93	0,85	0,86	0,85	0,83	0,85	0,86	0,82
c35	0,94	0,87	0,90	0,88	0,93	0,83	0,83	0,80
c36	0,96	0,95	0,99	0,99	0,91	0,96	0,99	0,96
c37	0,95	0,93	0,98	0,99	0,76	0,81	0,82	0,82
c38	0,95	0,94	0,97	0,98	0,77	0,81	0,85	0,86

Tabela 12 – Resultados de RTC e BD-Rate para as configurações propostas no de codificação normalizado, média dos vídeos HD 1080 e UHD 4K, para cada um dos quatro CQs analisados.

Config.	HD 1080		UHD 4K		Média		
	RTC (%)	BD-Rate (%)	RTC (%)	BD-Rate (%)	RTC (%)	BD-Rate (%)	Razão RTC/BD-Rate
c1	35,82	4,84	33,33	3,03	34,57	3,93	8,79
c2	11,22	0,40	1,12	0,29	6,18	0,34	17,97
c3	11,46	0,41	3,12	0,34	7,29	0,37	19,52
c4	13,68	0,35	7,01	1,07	10,35	0,71	14,52
c5	66,23	100,89	64,89	38,77	65,56	69,83	0,94
c6	40,60	34,00	41,55	10,92	41,07	22,46	1,83
c7	23,18	9,36	16,35	2,21	19,76	5,78	3,42
c8	11,67	0,55	0,18	0,00	5,92	0,28	21,49
c9	52,92	34,43	51,95	12,15	52,44	23,29	2,25
c10	33,51	9,86	26,86	3,31	30,18	6,59	4,58
c11	17,76	0,98	9,84	1,08	13,80	1,03	13,39
c12	15,02	0,44	9,895	1,08	12,45	0,76	16,41
c13	48,37	14,69	43,35	8,45	45,86	11,57	3,97
c14	34,09	5,65	23,44	6,11	28,76	5,88	4,89
c15	31,29	5,12	30,75	6,11	31,02	5,62	5,52
c16	59,06	24,81	60,73	22,35	59,89	23,58	2,54
c17	54,83	24,15	60,72	22,35	57,78	23,25	2,49
c18	76,72	42,05	82,04	41,54	79,38	41,80	1,90
c19	2,68	0,15	5,12	-0,02	3,07	0,07	47,05
c20	14,36	0,69	15,51	0,08	14,93	0,38	38,87
c21	6,42	0,20	5,96	0,29	6,19	0,25	24,95
c22	3,98	0,36	6,61	0,34	5,29	0,35	15,14
c24	3,59	0,30	3,93	0,05	3,76	0,17	21,55
c25	10,12	0,46	7,01	0,15	8,57	0,31	27,96
c26	1,85	0,23	2,49	-0,01	2,17	0,11	19,65
c27	0,58	0,08	1,81	0,01	1,20	0,05	26,25
c29	4,30	0,78	6,93	0,40	5,61	0,59	9,50
c30	40,42	4,68	46,61	2,52	43,51	3,60	12,10
c31	25,60	0,34	29,08	0,53	27,34	0,43	62,93
c32	15,08	0,16	17,65	0,24	16,36	0,20	83,01
c33	5,98	0,08	8,17	0,07	6,28	0,07	84,13
c34	12,66	0,86	19,04	0,76	15,08	0,81	18,68
c35	10,37	5,64	14,10	6,99	12,24	6,31	1,94
c36	2,76	3,86	5,69	0,64	4,23	2,251	1,88
c37	4,05	10,67	21,19	4,72	12,62	7,70	1,64

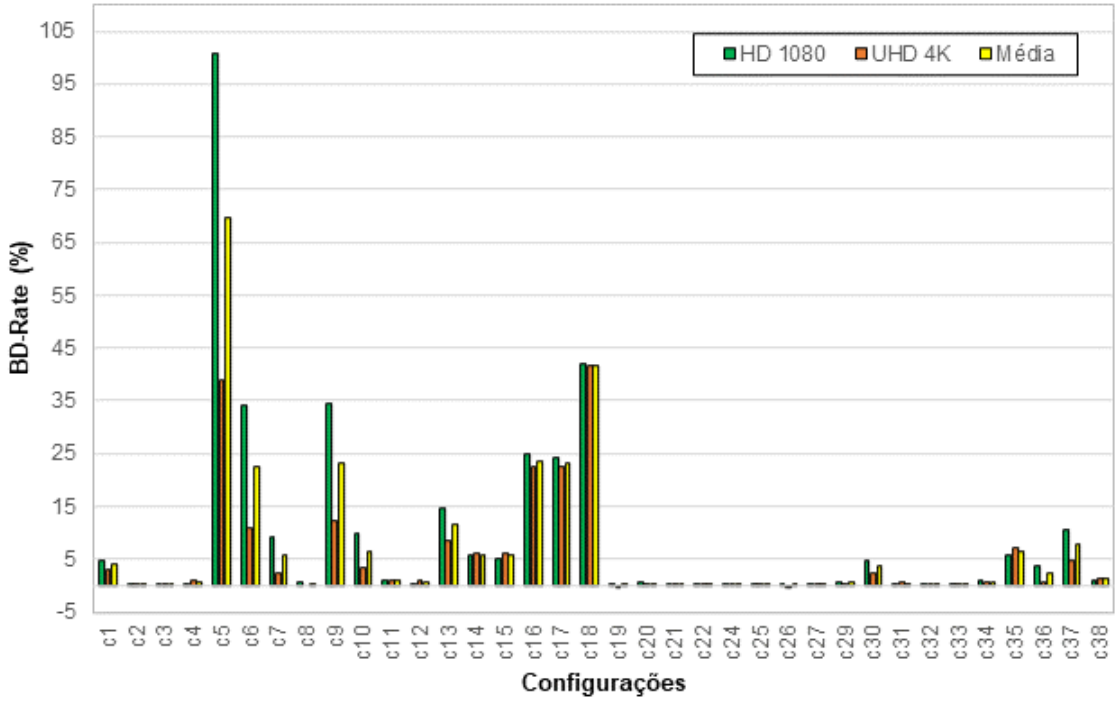


Figura 22 – Valores de BD-Rate para as configurações propostas.

ção no tempo médio total de codificação de, aproximadamente, 35%, com valor médio de BD-Rate de 3,93%.

As configurações *c4* e *c12* são semelhantes: ambas utilizam o tamanho mínimo de bloco 4×4 e limitam o tamanho máximo de bloco em 64×64. Com a configuração *c4* isso é obtido limitando o tamanho do SB em 64×64, já usando a configuração *c12* o tamanho mínimo e máximo do bloco são fixados, respectivamente, em 4×4 e 64×64. Na configuração *c4*, os valores relacionados ao custo taxa-distorção são calculados apenas a partir de blocos com tamanho 64×64, na configuração *c12*, alguns desses valores também são obtidos para blocos 128×128 e repassados para níveis inferiores da árvore de particionamento. Provavelmente, os algoritmos de terminação antecipada que o Libaom possui usam essas informações para evitar escolhas custosas em níveis inferiores. Isso pode ser observado nos resultados, onde a configuração *c4* reduziu em 10,35% o tempo médio total de codificação, com BD-Rate médio de 0,71%, já a configuração *c12* reduziu o tempo médio total de codificação 12,46%, com BD-Rate médio de 0,76%.

As configurações *c5* a *c18* limitam a profundidade da árvore de particionamento. Algumas delas promovem redução extremamente significativa no tempo de codificação, no entanto, também apresentam grande impacto na eficiência de codificação, como é o caso da configuração *c5.* Ela limita o codificador a usar apenas dois tamanhos de blocos (64×64 e 128×128) e reduz o tempo médio total de codificação em 65,56%. Em contrapartida, ela resulta em um valor de BD-Rate extremamente elevado: 69,83%. Situação semelhante ocorre com as configurações *c16* e *c17*, as

quais limitam o tamanho máximo de blocos em 16×16 e usam tamanho mínimo, respectivamente, 8×8 e 4×4. Apesar de contribuir positivamente na redução do tempo de codificação (cerca de 60%), apresentam valores de BD-Rate elevados, próximos de 23%. A configuração *c18*, que limita o tamanho máximo de blocos em 8×8 e tamanho mínimo de blocos em 4×4, também promove uma redução de tempo médio significativo (79,38%), porém o BD-Rate é ainda mais elevado 41,80%.

Nas configurações *c19* a *c34* são observados impactos causados por parâmetros usados pela predição inter-quadros. Entre elas, as três configurações que geraram maior redução no tempo médio de codificação foram a *c30*, a *c31* e a *c32*, com 43,51%, 27,34% e 16,36%, respectivamente. Elas alteram a quantidade máxima de quadros de referência permitidos para, respectivamente, 3, 4 e 5. O BD-Rate médio apresentado pela configuração *c30* foi de 3,60%, já as configurações *c31* e *c32* oferecem menores perdas de eficiência, 0,43% e 0,20%, respectivamente. Valores consideráveis de redução de tempo também podem ser observados nas configurações *c20* e *c34*, nas quais são desabilitadas as máscaras da predição composta e desabilitado o uso de vetores de movimento de outro quadro de referência para prever o vetor de movimento do bloco atual A redução média de tempo obtido na primeira foi de 14,93% com BD-Rate de 0,38%. A segunda reduziu, em média, 15,08% o tempo de codificação, com um BD-Rate de 0,81%.

Por fim, as configurações *c35* a *c38* modificam parâmetros relativos ao estágio de transformadas. Entre estas configurações, o valor mínimo de BD-Rate encontrado foi de 1,25% para a configuração *c38*, a qual usa apenas transformadas DCT na predição inter-quadros. Com esta configuração, o tempo médio de codificação foi reduzido em 12,10%.

Vale a pena mencionar que, para algumas configurações, os valores de BD-Rate e de redução de tempo são similares entre as duas resoluções analisadas, por exemplo, nas configurações *c15* e *c16*. No entanto, em muitos casos, os resultados das resoluções diferem, algumas vezes, de maneira bastante expressiva. Na configuração *c5*, por exemplo, para a resolução HD 1080 o valor de BD-Rate médio é próximo de 100%, já para a resolução UHD 4K esse valor é, aproximadamente, 40%. A significativa diferença entre as duas resoluções deve-se ao fato de que blocos grandes são mais representativos em resoluções maiores, modificando o comportamento dos codificadores, como já discutido na Seção 5.3.

As análises discutidas até aqui deram subsídio para o desenvolvimento do controlador apresentado nesta tese, pois a partir delas foram identificadas as etapas de codificação que deveriam ser exploradas pelo controlador, foi observada a importância da especialização do controlador de acordo com as características do vídeo de entrada e também foram identificados Pontos de Controle com potencial para serem usados no projeto do controlador. Os detalhes sobre o desenvolvimento do controlador

de complexidade serão apresentados no Capítulo 6.

6 CONTROLADOR DE COMPLEXIDADE ADAPTATIVO PARA O AV1

Neste capítulo, será apresentado o sistema de controle adaptativo desenvolvido para o codificador AV1. O controlador desenvolvido ajusta, dinamicamente, o ponto de operação de acordo com as características do vídeo de entrada para manter o tempo de codificação no alvo estipulado. A solução proposta é baseada nos experimentos e nas descobertas apresentadas no Capítulo 5, e será desenvolvida com base no diagrama simplificado mostrado na Figura 23. Como pode ser visto na Figura 23, o controlador proposto é composto por duas entradas: *Set Point* (SP), que é o valor alvo para a redução de complexidade, e tempo de codificação obtido pelo codificador AV1. O SP é uma entrada externa, inserida, por exemplo, por um usuário, ou por um monitor da carga da bateria, ou por um escalonador de tempo de processamento do sistema. A leitura do tempo de codificação é obtida a partir do software de referência, Libaom, e é realizada a cada quadro, a partir do momento que o codificador recebe o vídeo original. Tanto o tempo de codificação de cada quadro quanto o SP informado são usados para calcular o fator de redução de complexidade e, consequentemente, determinar os novos pontos de operação do codificador. O ponto de operação selecionado, obtido a partir de frente de Pareto, determina os parâmetros de codificação que devem ser habilitados/desabilitados, realizando, assim, o controle de complexidade do codificador.

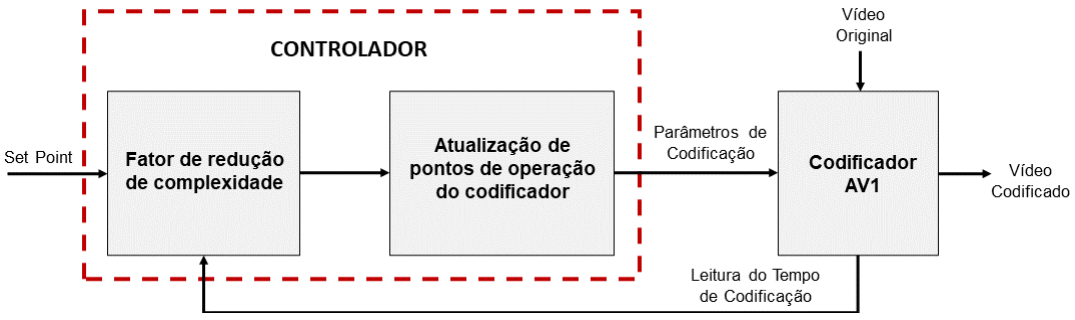


Figura 23 – Diagrama simplificado do sistema de controle proposto.

Para realizar a "prova de conceito" da viabilidade do controle adaptativo de com-

plexidade AV1, foi desenvolvida uma Versão *Baseline* do controlador, baseada em versões de controladores de complexidade implementados com foco em outros padrões de codificação de vídeo. Para fazer a Versão *Baseline*, muitos experimentos e decisões foram necessárias para adaptar o modelo de controlador ao codificador AV1. As principais decisões tomadas durante o desenvolvimento desta versão do controlador foram: a definição dos *PCs*, considerando valores médios de RTC e BD-Rate, e a definição da quantidade de quadros a ser usada como passo de controle.

A Versão *Baseline* foi aperfeiçoada a partir do desenvolvimento de um controlador otimizado e especializado, denominado *CCAM* (Controlador de Complexidade baseado em Aprendizado de Máquina). Os modelos de aprendizado de máquina se adaptam a novos dados, gerando decisões e resultados confiáveis em um curto espaço de tempo. Por isso, no controlador *CCAM* são utilizados dois modelos de aprendizado de máquina: um para realizar a predição do tempo de codificação e outro para classificar os vídeos de entrada.

A predição temporal é utilizada para estimar, com maior precisão, o tempo de codificação do próximo grupo de quadros a ser codificado e, consequentemente, selecionar o *PC* mais adequado. Já a classificação dos vídeos de entrada possibilita que o controlador aplique Pontos de Controle de acordo com as características dos respectivos vídeos. Dessa forma, os resultados do controlador são otimizados.

Para desenvolver o controlador *CCAM* vários experimentos foram necessários para auxiliar na tomada de decisão de diversos aspectos que impactam diretamente na eficiência do controlador, tais como: seleção dos conjuntos de teste, treino e validação para cada modelo, definição dos modelos de predição de tempo de codificação e classificação dos vídeos de entrada, e escolha dos métodos de implementação de cada um dos modelos desenvolvidos no controlador AV1.

Além de detalhar o desenvolvimento do controlador de complexidade adaptativo para o AV1, este capítulo apresenta conceitos básicos sobre sistema de controle (Seção 6.1) e frente de Pareto (Seção 6.2), assuntos pertinentes à compreensão da solução apresentada nesta tese.

6.1 Conceitos Básicos de Sistema de Controle

Um sistema de controle é a interconexão de componentes que gera uma configuração capaz de promover a resposta desejada ao sistema (DORF; BISHOP, 2017). A relação entrada e saída representa a relação de causa e efeito do processo, ou seja, há um processamento do sinal de entrada para fornecer um sinal de saída desejado (DORF; BISHOP, 2017). A literatura mostra que esta relação é representada, matematicamente, através da função de transferência. No entanto, a função de transferência de um codificador de vídeo é geralmente uma transformação não-linear e

complexa que é influenciada por muitos fatores, incluindo a configuração de codificação, a qualidade de entrada e as condições de transmissão. Além disso, o codificador pode realizar operações como quantização, filtragem, compactação e descarte de informações inúteis, tornando difícil determinar a sua função de transferência precisa. Portanto, levantar a função de transferência de um codificador de vídeo é, na prática, inviável.

Os sistemas de controle podem ser desenvolvidos em malha aberta ou malha fechada (*feedback*). O sistema de controle em malha fechada permite considerar a medida da atual resposta do sistema para efetuar o controle e, de acordo com Dorf; Bishop (2017), tem sido a base para a análise e projeto de sistemas de controle. O controle de *feedback* em malha fechada está vinculado ao conceito de sinal de erro, o qual consiste na diferença entre a saída desejada e a saída real. Além disso, o sistema de controle pode ser classificado como SISO (*Single Input Single Output*) ou MIMO (*Multiple Input Multiple Output*). No esquema SISO há apenas uma entrada e uma saída, já o MIMO é um sistema mais complexo, no qual pode haver diversas entradas e saídas.

A Figura 24 mostra um diagrama em blocos de um sistema SISO em malha fechada, similar ao utilizado nas soluções apresentadas nesta tese. O sensor "lê" a saída do processo, o qual representa a parte do sistema cujo comportamento se quer controlar e a medida de saída. Considerando um *feedback* positivo, este valor é somado ao valor de entrada e o resultado é usado pelo controlador para calcular as modificações que devem ser realizadas pelo atuador para levar o sistema ao comportamento desejado.

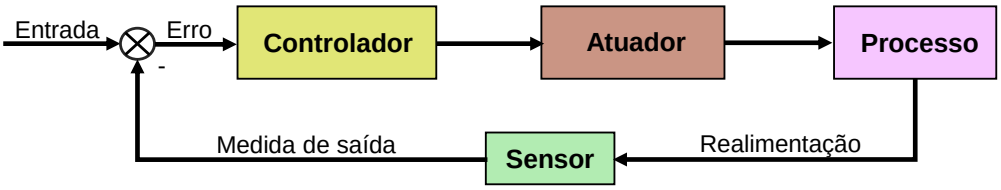


Figura 24 – Diagrama de blocos de um sistema de controle.

6.2 Frente de Pareto

Os problemas multiobjetivos, normalmente, contém objetivos conflitantes, sendo impossível determinar uma decisão que beneficie ambos objetivos. Para solucionar este tipo de problema, em 1896, o economista italiano Vilfredo Pareto propôs um método de otimização multiobjetivo conhecido como Eficiência à Pareto (GHOSH, 2004).

Segundo Ghosh (2004) a frente de Pareto é composta por pontos que são eficientes à Pareto e, geralmente, não apresenta como solução um único valor e sim um conjunto de valores. Tal conjunto é composto por pontos ótimos, pois, considerando

todos os objetivos, simultaneamente, não existem pontos melhores dentro do universo de busca.

No cenário de codificadores de vídeos, a busca por manter uma boa relação entre tempo de codificação e eficiência de codificação (BD-Rate) é desafiador, uma vez que ambos estão relacionados à configuração aplicada ao codificador durante o processo de codificação. Reduzir o tempo de codificação gera perdas na eficiência de codificação, ou seja, implica no aumento dos valores de BD-Rate. Dessa forma, este problema, comum para todos os codificadores, pode ser classificado como um problema multiobjetivo.

Cada configuração impacta de maneira distinta no tempo de codificação e no BD-Rate. Uma configuração é definida como parte da frente de Pareto se não houver outra configuração que a domine. Uma configuração dominante é aquela que alcança melhor BD-Rate e maior RTC do que a configuração em análise.

Na literatura científica atual podem ser encontradas soluções voltadas a codificação de vídeo que utilizam frente de Pareto. Em Corrêa (2014) e Penny et al. (2016) a frente de Pareto é usada para definir pontos de operação de um controlador de complexidade e um controlador de energia, respectivamente. Ambos aplicados ao padrão HEVC. O mesmo princípio também foi usado para definir a alocação de bits ótima para uma codificação escalável para os padrões H.264/SVC e HEVC (HWANG; LEE; PENG, 2019).

Diante disso, neste trabalho, os pontos ótimos de operação do controlador foram selecionados a partir do conceito de frente de Pareto aplicado às configurações analisas no Capítulo 5. Assim, as configurações utilizadas são as que proporcionam maior redução de tempo com o menor impacto possível na eficiência de codificação (BD-Rate).

6.3 Controlador Adaptativo de Complexidade para o AV1: Versão *Baseline*

Inicialmente, o controlador de complexidade para o AV1 desenvolvido nesta tese adaptou uma abordagem encontrada na literatura, utilizada tanto por Corrêa (2014) quanto por Penny et al. (2016) para controlar a complexidade e consumo de energia do codificador HEVC. O diagrama desta primeira versão de controlador adaptativo de complexidade para o AV1 pode ser observado na Figura 25 e será chamada de Versão *Baseline*. O pseudocódigo para este controlador é apresentado no Apêndice E desta tese.

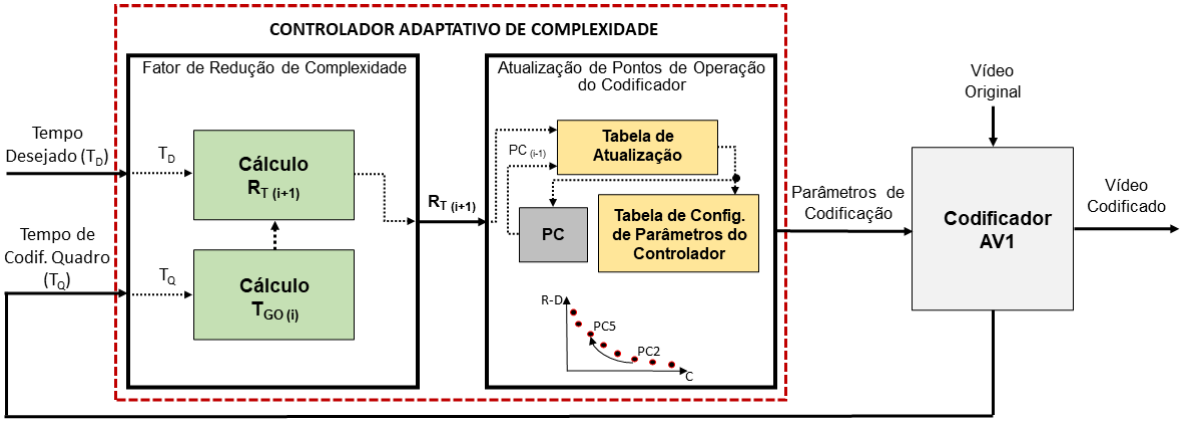


Figura 25 – Diagrama do controlador inicial de complexidade do AV1.

6.3.1 Definição dos Pontos de Controle

Os Pontos de Controle foram obtidos com base nos resultados da análise de sensibilidade das configurações mencionadas no Capítulo 5. Porém, há casos em que há incompatibilidade na combinação de algumas destas configurações, ou seja, não podem ser usadas simultaneamente pelo codificador e estas, naturalmente, não foram. Como exemplo, podem ser citadas as configurações *c5* a *c18*, as quais manipulam o nível de profundidade das árvores de particionamento, e as configurações *c30* a *c33*, usadas para definir a quantidade máxima de quadros de referência. Para estes casos, foram escolhidas as configurações nas quais a razão entre RTC e BD-Rate são mais favoráveis. Dessa forma, as configurações selecionadas foram: *c2*, *c3*, *c8*, *c19*, *c20*, *c21*, *c27*, *c29*, *c33*, *c34*, *c35*, *c36*, *c37* e *c38*.

Para selecionar as configurações com maior potencial de combinação foi utilizado, novamente, o conceito de frente de Pareto, mencionado na Seção 6.2. Como mostrado na Figura 26 as configurações selecionadas a partir da frente de Pareto foram: *c3*, *c19*, *c20*, *c27*, *c33* e *c34*. No entanto, os parâmetros *dist-wtd-comp*, *masked-comp* e *sb-size* estão localizados em uma estrutura diferente dos demais parâmetros. Essa estrutura é definida pelo Libaom a cada sequência de vídeo, não sendo possível alterá-las durante a codificação. Logo, as configurações *c19*, *c20* e *c4*, relacionadas aos parâmetros citados, foram descartadas das demais análises.

A fim de representar uma única configuração ou a combinação das configurações *c3*, *c27*, *c33* e *c34*, e obter níveis específicos de redução de tempo de codificação, foram criados os conjuntos de parâmetros (*CP*). Os *CPs* foram nomeados respeitando a ordem crescente de RTC. O *CP0*, vinculado à configuração *c0*, representa a referência para os demais, por utilizar as configurações padrão do codificador Libaom. Além do *CP0*, outros 44 *CPs* foram definidos. Os resultados médios referentes à RTC e BD-Rate de cada *CP* podem ser observados na Tabela 13, dados estes que também são apresentados, graficamente, na Figura 27.

À medida que o índice CP aumenta, o impacto na redução do tempo de codificação

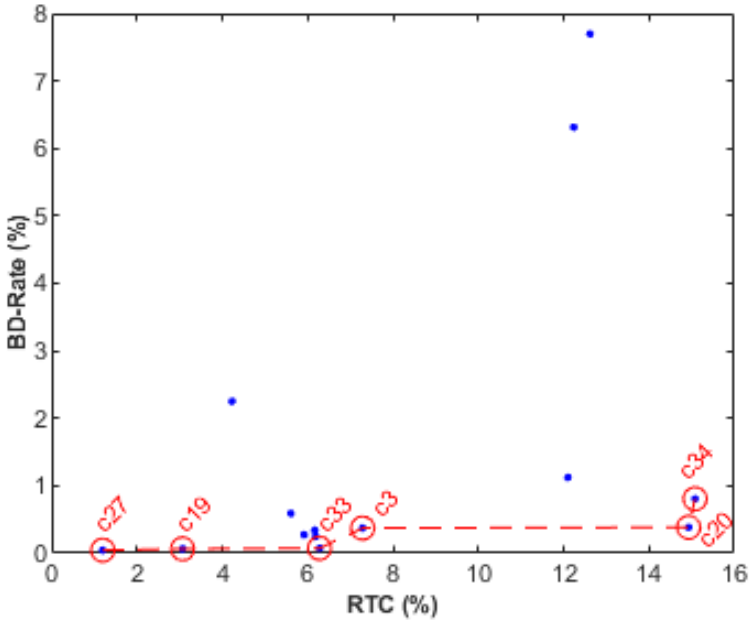


Figura 26 – Frente de Pareto com as configurações selecionadas para combinação.

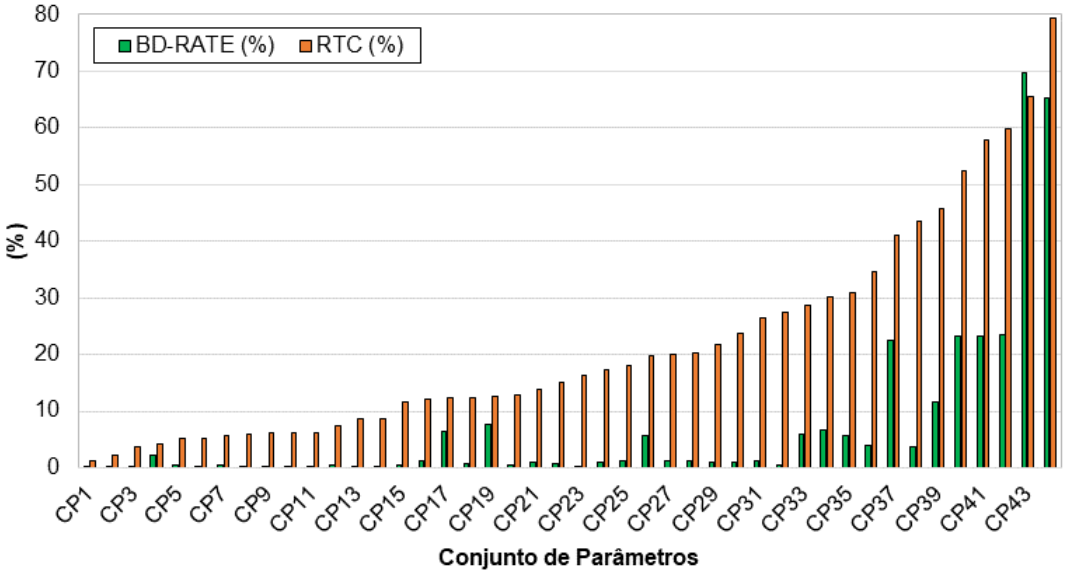


Figura 27 – Redução de Tempo de Codificação (RTC) e BD-Rate para cada Conjunto de Parâmetros.

Tabela 13 – Conjunto de Parâmetros propostos.

Conjunto de Parâmetros	Configurações	RTC (%)	BD-Rate (%)
CP1	c27	1,20	0,05
CP2	c26	2,17	0,11
CP3	c24	3,76	0,17
CP4	c36	4,23	2,25
CP5	c27, c3	5,20	0,39
CP6	c22	5,29	0,35
CP7	c29	5,61	0,59
CP8	c8	5,92	0,28
CP9	c2	6,17	0,34
CP10	c21	6,19	0,25
CP11	c33	6,28	0,07
CP12	c3	7,29	0,37
CP13	c27, c33	8,55	0,11
CP14	c25	8,57	0,31
CP15	c33, c3	11,70	0,42
CP16	c38	12,10	1,12
CP17	c35	12,24	6,31
CP18	c12	12,45	0,76
CP19	c37	12,62	7,70
CP20	c27, c33, c3	12,85	0,45
CP21	c11	13,80	1,03
CP22	c34	15,08	0,81
CP23	c32	16,36	0,20
CP24	c27, c34	17,23	0,89
CP25	c3, c34	18,10	1,13
CP26	c7	19,76	5,78
CP27	c27, c3, c34	20,10	1,21
CP28	c33, c3, c34	20,38	1,24
CP29	c33, c34	21,77	0,90
CP30	c27, c33, c34	23,83	0,97
CP31	c27, c33, c3, c34	26,47	1,31
CP32	c31	27,34	0,43
CP33	c14	28,76	5,88
CP34	c10	30,18	6,59
CP35	c15	31,02	5,62
CP36	c1	34,57	3,93
CP37	c6	41,07	22,46
CP38	c30	43,51	3,60
CP39	c13	45,86	11,57
CP40	c9	52,44	23,29
CP41	c17	57,78	23,25
CP42	c16	59,89	23,58
CP43	c5	65,56	69,83
CP44	c18	79,38	65,30

é acrescido. O conjunto de parâmetros proposto reduziu o tempo de codificação entre 1,20% e 79,38%. No entanto, como já foi mencionado, avaliar apenas o impacto da redução do tempo de codificação não é recomendável, pois a análise deve considerar também as consequências trazidas para a eficiência de codificação. Também é possível observar uma variação bastante significativa no BD-Rate entre os diferentes CPs: entre 0,05% e 69,83%.

A redução de tempo de codificação traz impactos diversos na eficiência de codificação. Em alguns casos, os impactos são expressivos, enquanto em outros, os impactos são menores. Assim, para encontrar os pontos ótimos de operação do controlador, ou seja, pontos constituídos por conjunto de parâmetros que apresente uma relação favorável entre redução de tempo e eficiência de codificação, foi utilizado novamente o conceito de frente de Pareto. Dessa forma, foram selecionados pontos que proporcionam maior redução de tempo com o menor impacto possível na eficiência de codificação.

Os 44 CPs apresentados foram usados para construir a frente de Pareto representada pela linha tracejada vermelha na Figura 28, que relaciona redução de tempo e BD-Rate.

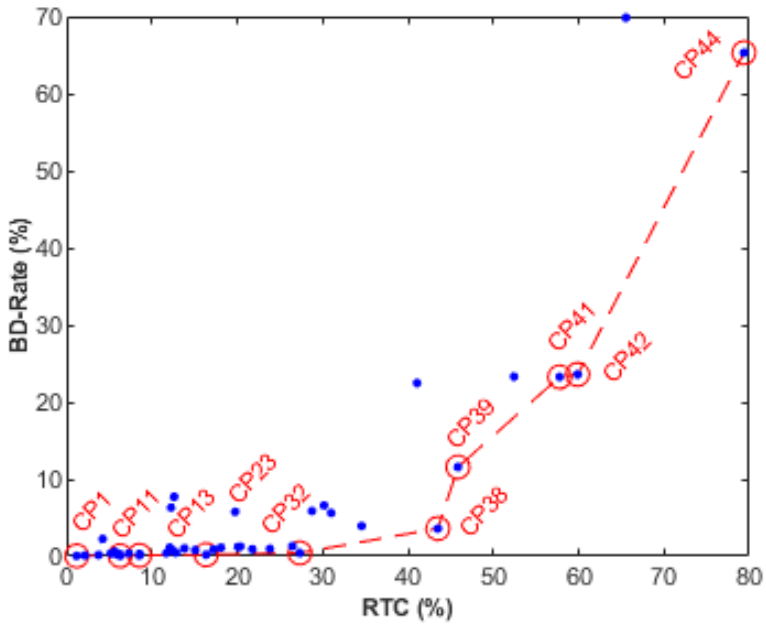


Figura 28 – Frente de Pareto relacionando valores médios de redução do tempo de codificação e BD-Rate dos *CPs*.

Os *CPs* que pertencem à frente de Pareto estão circulados e os respectivos índices destacados. É possível observar que alguns *CPs* correspondem a faixas de RTC muito próximas, como o *CP11* e o *CP13*. Ambos se encontram na faixa de RTC de 10%. Assim, para tratar esses casos, mantendo apenas um Ponto de Controle para cada faixa de RTC, uma segunda análise considerando informações de BD-Rate, redução

de tempo de codificação (RTC), razão entre essas duas métricas e desvio padrão (DP) foi realizada.

A metodologia de seleção, aplicada exclusivamente nos pontos que promovem redução no tempo de codificação, pode ser observada no fluxograma da Figura 29, o qual é composto pelos seguintes termos:

- PC_x: Ponto de Controle em avaliação.
- PC_{x+1}: Próximo Ponto de Controle a ser avaliado.
- BD-Rate_PC_x_vídeo$_y$: valor de BD-Rate do Ponto de Controle em avaliação para uma sequência de vídeo específica.
- razão_PC_x_vídeo$_y$: valor da razão (relação entre os valores de RTC e BD-Rate) do Ponto de Controle em avaliação para uma sequência de vídeo específica.
- razão$_M$: maior valor de razão, entre as sequências analisadas com a mesma resolução, referente ao Ponto de Controle em avaliação.

Primeiramente, os dados foram tratados (etapas em azul, na Figura 29). Depois, critérios mais minuciosos de seleção foram aplicados. Eles estão representados pelas etapas em amarelo na Figura 29 e foram definidos empiricamente.

Considerando o fato que o BD-Rate está presente no denominador do cálculo da razão, valores de BD-Rate muito baixos geram valores de razão elevados. Porém, na prática, um BD-Rate de 0,1% e outro de 0,01% impactam de maneira similar na qualidade da sequência de vídeo. Por isso, para cada sequência de vídeo, são atribuídos pesos para a razão de acordo com os valores de BD-Rate. A razão relacionada aos pontos com BD-Rate negativo ou zero, recebe o maior valor de razão entre todos os pontos observados, garantindo, assim, que a razão não seja excessivamente impactada por estes valores de BD-Rate. Se o *PC* apresentar algum valor de BD-Rate zero ou negativo e for o único na faixa de RTC analisada, ele é diretamente selecionado. Caso exista mais de um ponto na faixa de redução de tempo que está sendo analisada, utiliza-se como critério de desempate o valor da razão. Se a diferença entre as razões dos pontos em análise for menor que 2%, então é selecionado o ponto que apresenta menor desvio padrão em relação ao valor médio da razão. Pontos com menor desvio padrão tendem a ser mais interessantes, pois são mais próximos ao valor médio da razão. Para evitar a seleção de *PCs* com valores de RTC muito semelhantes, o próximo ponto a ser escolhido deve apresentar RTC maior ou igual a 3% em relação ao último ponto selecionado.

O fluxo de seleção é semelhante para os pontos com e sem valores de BD-Rate negativo ou zero. A única diferença é que para os pontos com valor de BD-Rate negativo ou zero, a seleção pode ocorrer independentemente do valor de RTC apresentado. Já

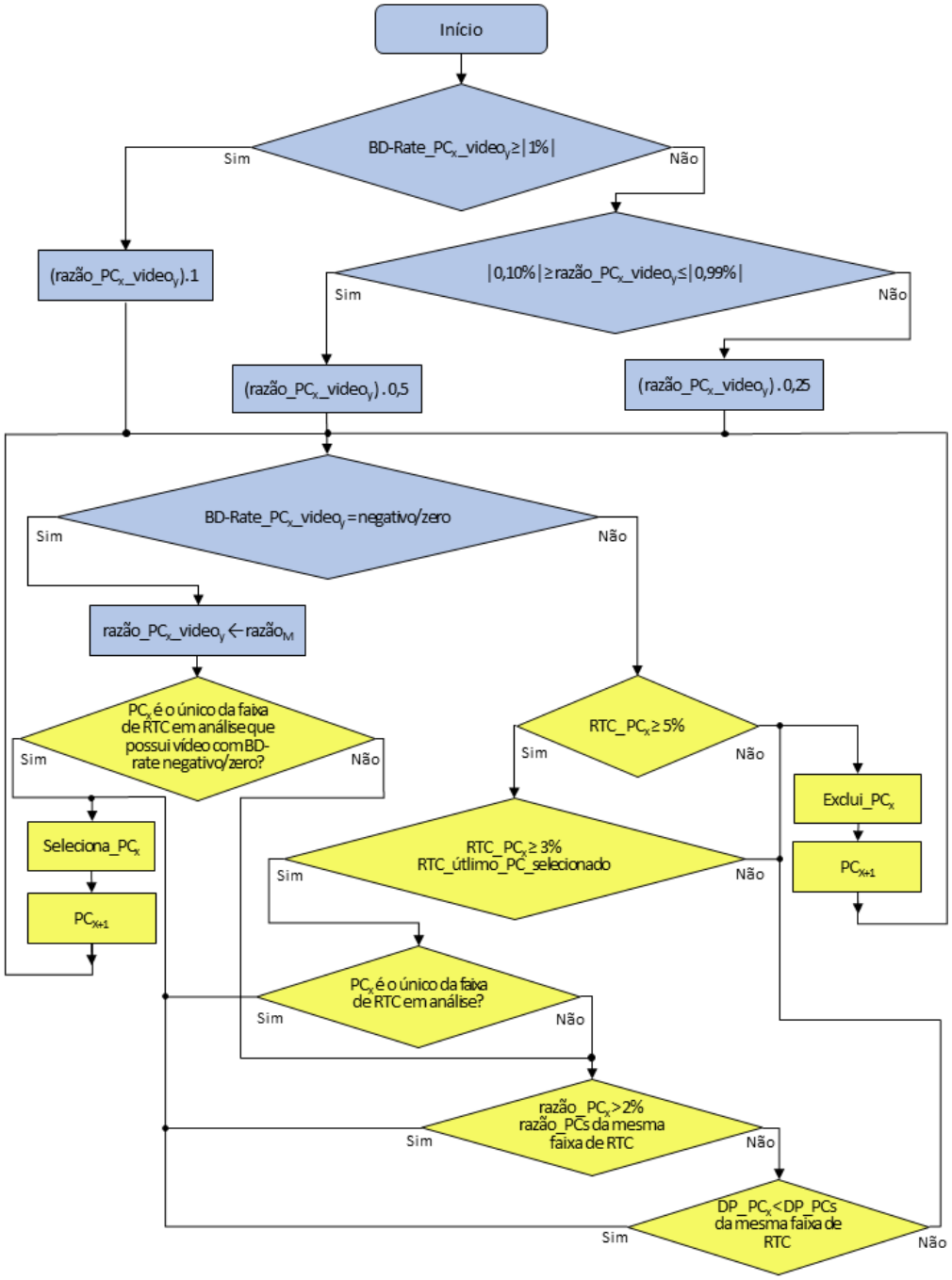


Figura 29 – Fluxograma para seleção dos PCs.

para os pontos que não se enquadram neste caso (BD-Rate não é negativo ou zero), a seleção só é permitida caso a RTC seja de, no mínimo, 5%, critério este também adotado por Corrêa (2014).

Dessa forma, a partir da metodologia descrita, os Pontos de Controle *PC* foram selecionados. No Apêndice D é possível observar os Pontos de Controle escolhidos para cada uma das sequências de vídeo analisadas e um exemplo da metodologia de seleção sendo aplicada passo a passo. A Tabela 14 mostra o ponto de referência *PC0* e os sete Pontos de Controle selecionados. A partir desses pontos, o controlador desenvolvido promoverá o controle adaptativo da complexidade do codificador AV1. Também são apresentados os *CPs* que caracterizam cada PC e as respectivas configurações atribuídas. Além disso, para cada *PC*, podem ser observados o tempo de codificação normalizado, redução de tempo e BD-Rate.

Tabela 14 – Pontos de Controle definidos para o controlador.

Pontos de Controle	Conjunto de Parâmetros	Configurações	Tempo de Codificação Normalizado	RTC (%)	BD-Rate (%)
PC0	CP0	c0	1	0	0
PC1	CP1	c27	0,99	1,20	0,05
PC7	CP13	c27, c33	0,91	8,55	0,11
PC12	CP23	c32	0,84	16,36	0,20
PC15	CP32	c31	0,73	27,34	0,43
PC20	CP38	c30	0,56	43,51	3,60
PC24	CP42	c16	0,40	59,89	23,58
PC26	CP44	c18	0,21	79,38	65,30

Pode-se observar que, para as sequências de vídeos testadas, os Pontos de Controle selecionados reduzem a complexidade de 1,20% a 79,38%. Já o BD-Rate varia de 0,05% a 65,30%. Em Zhang et al. (2018), por exemplo, é realizada uma comparação entre diversas soluções de controle aplicadas ao HEVC usando configuração RA (*Random Acess*). Os valores médios obtidos para BD-Rate, considerando redução de complexidade de 40%, variaram de 2,10% a 25,27%, logo, proporcionar uma redução de complexidade de 43,51% a custo de um BD-Rate de 3,6%, caso do *PC20*, pode ser considerado um bom resultado. Já para a redução de complexidade de 60%, os valores médios de BD-Rate se encontram na faixa de 7,09% a 35,52%. O *PC24* atinge uma redução de complexidade de 59,89% a custo de um BD-Rate de 23,58%. Por fim, na Tabela 15 são apresentados os valores atribuídos aos parâmetros relacionados a cada *PC*.

Na Tabela 16 são apresentadas as variações dos valores normalizados da redução de complexidade com a alteração entre os *PCs*. Além do *keyframe*, os primeiros 16 quadros de cada vídeo são sempre codificados com *PC0*, ou seja, a configuração padrão é utilizada como âncora para decisão do PC do próximo grupo, no qual de fato o controlador passa a atuar. Cada coluna representa o Ponto de Controle anterior.

Tabela 15 – Parâmetros atribuídos para cada PC utilizado pelo controlador.

Parâmetros	**PC0**	**PC1**	**PC7**	**PC12**	**PC15**	**PC20**	**PC24**	**PC26**
onesided-comp	1	1	1	1	1	1	1	1
interintra-comp	1	1	1	1	1	1	1	1
diff-wtd-comp	1	1	1	1	1	1	1	1
interinter-wedge	1	1	1	1	1	1	1	1
interintra-wedge	1	1	1	1	1	1	1	1
global-motion	1	0	0	1	1	1	1	1
obmc	1	1	1	1	1	1	1	1
max-reference-frames	7	7	6	5	4	3	7	7
ref-frames-mvs	1	1	1	1	1	1	1	1
tx64	1	1	1	1	1	1	1	1
flip-idtx	1	1	1	1	1	1	1	1
reduced-tx-type-set	0	0	0	0	0	0	0	0
use-inter-dct-only	0	0	0	0	0	0	0	0
rect-partitions	1	1	1	1	1	1	1	1
ab-partitions	1	1	1	1	1	1	1	1
1to4-partitions	1	1	1	1	1	1	1	1
min-partition-size	4	4	4	4	4	4	8	4
max-partition-size	128	128	128	128	128	128	16	8

As linhas são percorridas a fim de encontrar o valor igual ou menor mais próximo ao $R_{T_{(i+1)}}$ calculado e, assim, selecionar o novo Ponto de Controle e as configurações que devem ser repassadas ao codificador. Este processo é repetido, sucessivamente, até todos os quadros do vídeo serem codificados. Caso o valor de $R_{T_{(i+1)}}$ seja menor que a menor complexidade disponível na tabela de transição, o *PC* máximo é selecionado.

Tabela 16 – Tabela de atualização do controlador.

Ponto de Controle		**Anterior**							
		PC0	**PC1**	**PC7**	**PC12**	**PC15**	**PC20**	**PC24**	**PC26**
Próximo	PC0	**1,00**	1,01	1,10	1,19	1,37	1,79	2,50	4,76
	PC1	0,99	**1,00**	1,09	1,18	1,36	1,77	2,48	4,71
	PC7	0,91	0,92	**1,00**	1,08	1,25	1,63	2,28	4,33
	PC12	0,84	0,85	0,92	**1,00**	1,15	1,50	2,10	4,00
	PC15	0,73	0,74	0,80	0,87	**1,00**	1,30	1,83	3,48
	PC20	0,56	0,57	0,62	0,67	0,77	**1,00**	1,40	2,67
	PC24	0,40	0,40	0,44	0,48	0,55	0,71	**1,00**	1,90
	PC26	0,21	0,21	0,23	0,25	0,29	0,38	0,53	**1,00**

Se o valor de $R_{T_{(i+1)}}$ for igual ou menor que 1, isto significa que o tempo observado é igual ou maior que o tempo desejado e se faz necessária a redução de complexidade, ou seja, *PCs* que resultem em maior redução do tempo devem ser escolhidos. Caso o tempo observado seja menor que o tempo desejado, então o controlador pode promover o aumento de complexidade, para o próximo grupo de quadros e, neste caso, pode selecionar *PCs* que apresentem menores valores de RTC, no entanto, com ganhos em BD-Rate. O controlador pode migrar entre os *PCs* sem restrições. Se, por

exemplo, no *PC0* o tempo médio de codificação obtido foi de 0,5s, e é desejado um tempo de codificação de 0,4s (redução de 20%), o novo Ponto de Controle passa a ser *PC15*. Caso seja requerida outra redução de complexidade, o controlador percorrerá novamente a tabela, agora tendo como base a coluna *PC15*, para encontrar o novo Ponto de Controle.

6.3.2 Definição do Passo de Controle

Apesar da leitura do tempo de codificação ocorrer por quadro, para fins de decisão do controlador é mais recomendável agrupar os tempos por grupos de quadro, pois evita variação excessiva do sistema de controle. No entanto, o AV1 usa GFGs dinâmicos, ou seja, o tamanho do GFG pode variar no decorrer da codificação, dificultando a definição do passo de controle. Diante disso, foi realizada uma análise para identificar a quantidade de ocorrência dos tamanhos de GFGs permitidos e decidir a quantidade de quadros usada para as decisões do controlador, isto é, o passo de controle. Para os quatro CQs recomendados (DAEDE; NORKIN; BRAILOVSKIY, 2020), foram observadas 24 sequências de vídeos, presentes em (XIPH, 2022b), com resoluções HD 1080 e UHD 4K. Essas sequências são listadas no Apêndice C desta tese.

Baseado nos resultados apresentados na Figura 30, e também observados na Tabela 17, é possível afirmar que o GFG composto por 16 quadros é o mais utilizado. Este tamanho de GFG foi observado em cerca de 73% dos casos analisados. Os outros quatro tamanhos de GFG usados com maior frequência foram 3, 4, 8 e 10, representando, respectivamente, 8,32%, 2,16%, 3,85% e 3,24% dos casos.

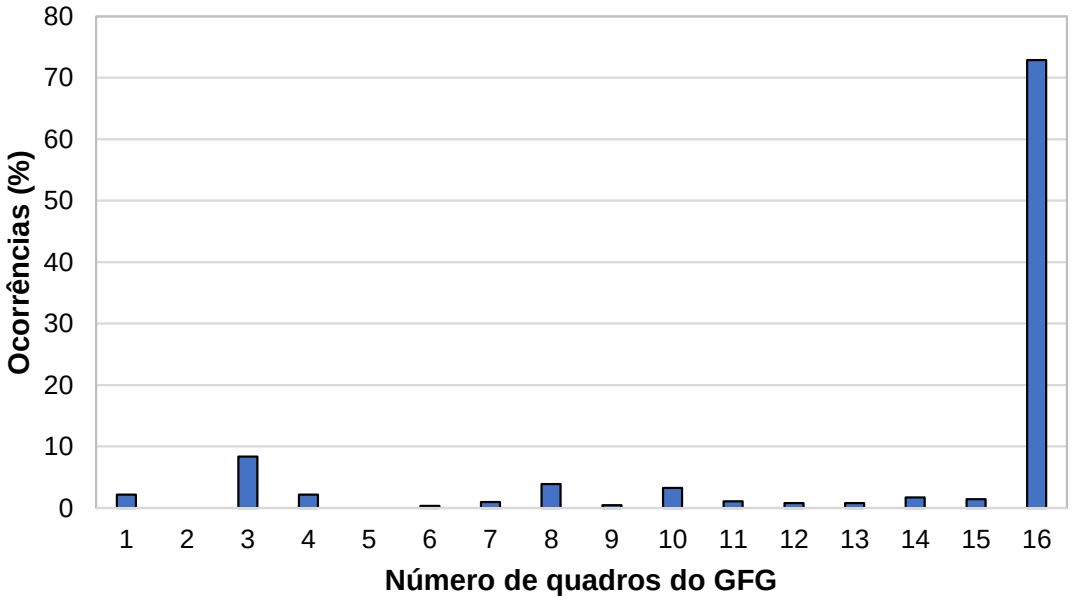


Figura 30 – Gráfico com a distribuição de ocorrência para diferentes tamanhos de GFG.

O GFG com 16 quadros apresenta um comportamento cíclico, que pode ser observado nos resultados de tempo de codificação para a sequência de vídeo *Net-*

Tabela 17 – Percentual médio de ocorrências para diferentes tamanhos de GFG.

Tamanho GFG	Ocorrências (%)
1	2,16
2	0,00
3	8,32
4	2,16
5	0,00
6	0,31
7	0,92
8	3,85
9	0,46
10	3,24
11	1,08
12	0,77
13	0,77
14	1,69
15	1,39
16	72,88

flix_SaquareAndTimelapse, usando o CQ = 55, Figura 31. Nota-se que, do quadro 2 ao quadro 52, o tempo de codificação segue um padrão que se repete de 16 em 16 quadros. Este padrão deixa de ser observado do quadro 53 ao 60. Cabe salientar que o primeiro quadro é o *keyframe*, ou seja, é um quadro intra.

A fim de verificar o impacto na RTC promovido por cada um dos cinco tamanhos de GFG mais utilizados (3, 4, 8, 10 e 16), esses valores foram aplicados como passo de controle na Versão *Baseline* do controlador. Os experimentos consideraram os quatro CQs recomendados (DAEDE; NORKIN; BRAILOVSKIY, 2020) e alvo de RTC de 10%, 20%, 30% e 40%. Foram observados os 180 quadros de três sequências de vídeos, com resolução HD 1080, que usam diferentes assinaturas de GFG. Os resultados médios, apresentados na Tabela 18, mostram quantas vezes determinado passo de controle promoveu a melhor precisão de codificação. Além disso, é possível observar os tamanhos de GFG utilizados durante a codificação das sequências de vídeo analisadas. A coluna "outros" une os resultados obtidos a partir dos passos de controle 3, 4 e 10.

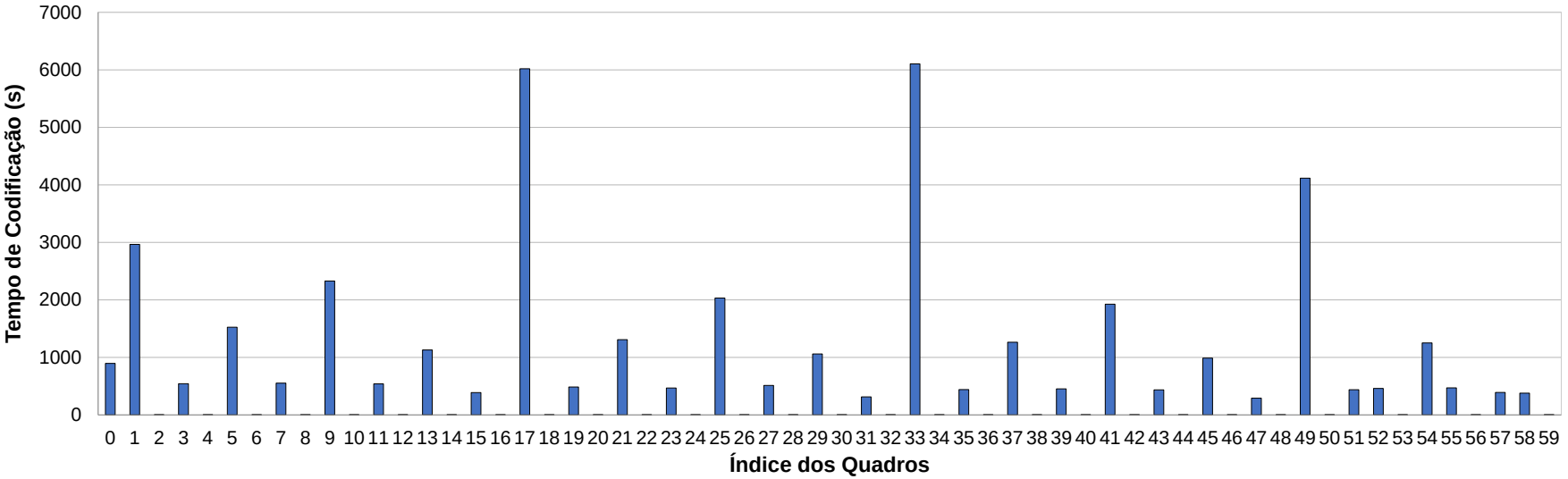


Figura 31 – Tempo de codificação de cada quadro da sequência de vídeo *Netflix_SaquareAndTimelapse*, CQ = 55.

Tabela 18 – Ocorrência de melhor precisão do controlador AV1 com diferentes quantidades de quadros sendo usados como passo de controle.

Sequência de Vídeo	**Tamanho do GFG**	**Ocorrência de Melhor Precisão**		
		Passo de Controle 8	**Passo de Controle 16**	**Passo de Controle "outros"**
station2_1080p25	altera entre 8 e 10	56%	31%	13%
pan_hdr_amazon_1080p	16	6%	88%	6%
seaplane_hdr_amazon_1080p24	primeiro 12 restantes 16	19%	75%	6%

Para as sequências *pan_hdr_amazon_1080p* e *seaplane_hdr_amazon_1080p24*, o uso do passo de controle de 16 em 16 quadros gerou melhor precisão na grande maioria dos casos analisados, 88% e 75%, respectivamente. Este foi o passo de controle mais expressivo tanto para a sequência que usa, exclusivamente, GFG com 16 quadros (*pan_hdr_amazon_1080p*) quanto para a sequência com primeiro GFG = 12 e os restantes iguais a 16 (*seaplane_hdr_amazon_1080p24*). Cabe salientar que neste último caso, o fato do primeiro GFG ser igual a 12, produz um deslocamento na assinatura correspondente a 16 quadros.

Além disso, mesmo na sequência de vídeo que não utiliza tamanho de GFG igual a 16 (o tamanho de GFG alterna entre 8 e 10) o uso do passo de controle de 16 em 16 quadros promoveu melhor precisão de controle em uma quantidade significativa de casos, 31%, perdendo apenas para o passo de controle de 8 em 8, o qual atinge a melhor precisão para 56% dos casos.

Assim, diante dos resultados observados e da necessidade de fixar o tamanho do passo de controle para viabilizar a implementação do controlador, definiu-se tomar as decisões de controle considerando os tempos de codificação de grupos compostos por 16 quadros, como mostra a Equação 7. O tempo de codificação observado para o grupo de quadros, $T_{GO_{(i)}}$, é obtido a partir do somatório do tempo de codificação de cada quadro $T_{Q_{(x)}}$. Onde i corresponde ao índice do grupo de quadros que está sendo observado e x é o índice de cada quadro. A Equação 7 é aplicável para todos os grupos de quadros, exceto para o grupo 1, no qual esta contido o *keyframe*.

$$T_{GO_{(i)}} = \sum_{x=16(i-1)+1}^{16i+1} T_{Q_{(x)}} \tag{7}$$

A partir da RTC alvo, o usuário determina o tempo de codificação desejado (T_D) com base na Equação 8, e informa ao controlador. Na Equação 8, p corresponde a redução do tempo de codificação desejada em porcentagem, T_{CP} é o tempo de codificação obtido com a configuração padrão (codificador AV1 original) e N_Q é o

número de quadros da sequência de vídeo.

$$T_D = \frac{(16.p.T_{CP})}{(100.N_Q)} \tag{8}$$

Posteriormente, o sinal de controle $R_{T_{(i+1)}}$ é calculado, a partir da Equação 9. O $R_{T_{(i+1)}}$ corresponde a relação entre o tempo de codificação desejado (T_D) e o tempo de codificação do último grupo de quadros ($T_{GO_{(i)}}$). Caso seja necessário ajustar a complexidade (tempo observado é maior ou menor que o tempo desejado), com base na tabela de atualização (Tabela 16), o controlador irá atuar e buscar o próximo ponto de operação, também tratados neste trabalho como Pontos de Controle (*PC*).

$$R_{T_{(i+1)}} = \frac{T_D}{T_{GO_{(i)}}} \tag{9}$$

A Versão *Baseline* apresentada foi utilizada como ponto de partida para averiguar a possibilidade de desenvolver um controlador adaptativo de complexidade focado no formato AV1, dada a inexistência, na literatura especializada atual, de um controlador de complexidade para o codificador AV1. Porém, alguns pontos negativos, discutidos a seguir, são observados nesta versão do controlador.

A Versão *Baseline* utiliza Pontos de Controle obtidos a partir de resultados médios de RTC e BD-Rate de um conjunto de sequências de vídeos com resolução HD 1080 e UHD 4K. No entanto, estes pontos podem não atender, de maneira eficiente, a determinada sequência de vídeo de entrada, pois, em geral, as sequências de vídeos diferem em diversos aspectos, como na textura, no tipo e intensidade de movimento, etc.

Outro fator relevante é que a seleção do Ponto de Controle é realizada apenas com base na relação entre o tempo desejado (T_D) e o tempo de codificação referente ao último grupo de quadros ($T_{GO_{(i)}}$). Não está sendo aplicado nenhum mecanismo que considere o *budget* de tempo, ou seja, a diferença entre o tempo desejado e o tempo de codificação dos grupos de quadro anteriores. Dessa forma, mesmo que haja "saldo" ou "déficit" de complexidade para ser explorado, isso não é feito. Por exemplo, se o tempo alvo de codificação de um determinado grupo é 100s e ele atinge 90s, os 10s restantes estão sendo ignorados na seleção do próximo Ponto de Controle. Se neste mesmo caso, o tempo desejado de codificação do próximo grupo de quadros fosse acrescido em 10s, provavelmente, o Ponto de Controle selecionado seria mais adequado, impactando positivamente na precisão do controlador.

Além disso, a decisão do próximo *PC* a ser selecionado está desconsiderando possíveis diferenças de tempo de codificação entre os grupos de quadros da sequên-

cia de entrada. Porém, as sequências de vídeo apresentam tempos distintos de codificação para cada quadro e, consequentemente, para cada grupo de 16 quadros, como pode ser observado na Figura 32. A Figura 32 apresenta o tempo de codificação da sequência *ducks_take_off_1080p50_60f*, resolução HD 1080, e da sequência *Neon1224_3840x2160_2997fps_180f*, resolução UHD 4K, ambas codificadas com CQ = 20. Os dados mostram que a diferença entre o grupo 1 e 2 (quadros 1 a 32) da sequência ducks_take_off_1080p50_60f, corresponde a 11%. Já para a sequência Neon1224_3840x2160_2997fps_180f este valor é, aproximadamente, 14%. Cabe destacar que, nesta sequência de vídeo, a diferença entre os tempos de codificação dos grupos 10 e 11 (quadros 160 a 176) é ainda mais expressiva, aproximadamente, 30%.

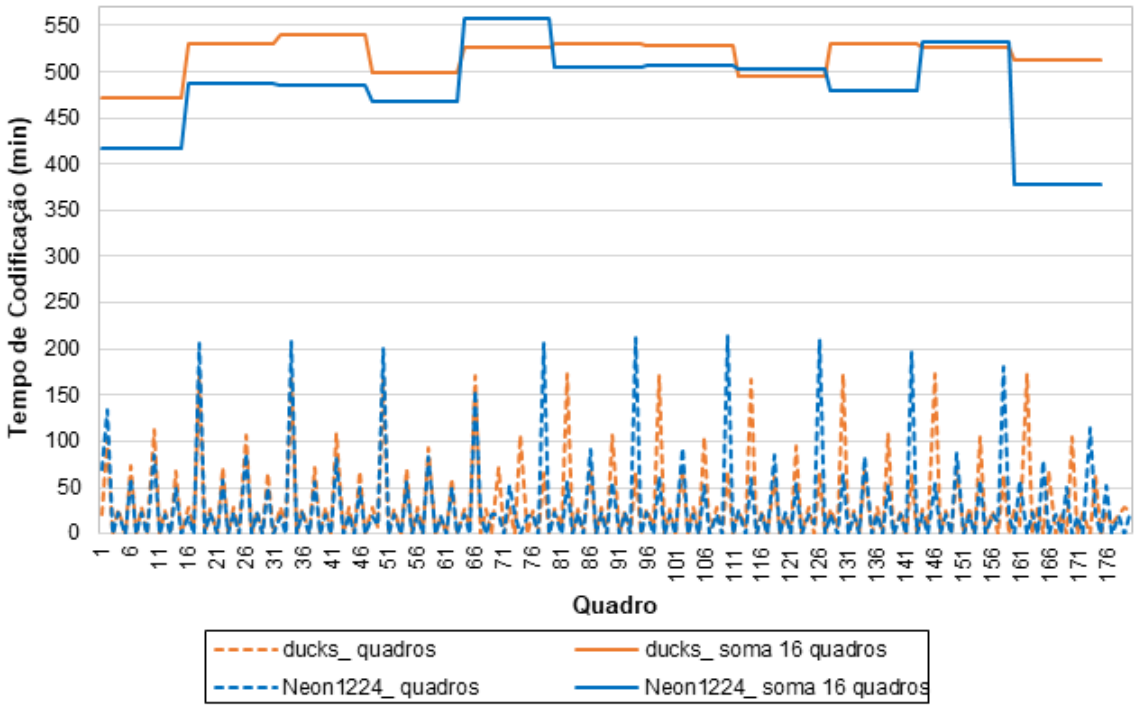


Figura 32 – Tempo de codificação com a configuração padrão usando o CQ 20 para as sequências de vídeo *ducks_take_off*, resolução HD 1080 (laranja) e *Neon1224*, resolução UHD 4K (azul).

6.4 Controlador de Adaptativo de Complexidade para o AV1 baseado em Aprendizado de máquina

A Figura 33 apresenta o diagrama do controlador de complexidade adaptativo para o AV1 baseado em aprendizado de máquina, chamado de *CCAM*. As melhorias/inovações apresentadas por este controlador em relação a Versão *Baseline* estão listadas a abaixo:

- Predição do tempo de codificação;

- Especialização dos Pontos de Controle;
- Utilização do *budget* de tempo para a seleção dos Pontos de Controle.

A Versão *Baseline* assume que o tempo de codificação entre o grupo de quadros anterior e o atual é, exatamente, o mesmo. No controlador *CCAM*, isto é corrigido com o uso da predição do tempo de codificação, a qual considera diferenças de tempo de codificação entre os grupos de quadros da sequência de entrada. Além disso, a Versão *Baseline* aplica um único conjunto de *PCs*, independentemente das características do vídeo a ser codificado. Porém, dar ao controlador a capacidade de especialização, possibilitando o uso *PCs* apropriados a cada vídeo de entrada, pode melhorar os resultados de controle.

O controlador *CCAM* realiza a predição do tempo de codificação e a especialização dos Pontos de Controle com base em modelos de aprendizado de máquina, ferramenta ideal para auxiliar o controlador a tomar decisões que promovam o controle de complexidade de maneira eficiente. Além disso, diferentemente da Versão *Baseline*, o controlador *CCAM* considera o *budget* de tempo disponível para escolher os Pontos de Controle, explorando, assim, "saldos" e "défices" de complexidade dos grupos anteriores de quadros.

As informações estatísticas de cada quadro do vídeo a ser codificado, geradas na primeira passada de codificação Libaom, são coletadas e usadas pelos algoritmos de aprendizado de máquina sem que o vídeo de entrada sofra processamento adicional. Essas informações, identificadas nas estruturas FIRST_PASS_STATS e FRAME_STATS do codificador, são usadas como atributos dos modelos desenvolvidos para realizar a predição do tempo de codificação e a seleção dos Pontos de Controle a partir das características do vídeo de entrada. Ambos modelos foram desenvolvidos a partir das seguintes etapas: treinamento, teste e validação.

O controlador *CCAM* é constituído por duas etapas (Figura 33), denominadas: Fator de Redução de Complexidade (FRC) e Atualização de Pontos de Operação do Codificador (APOC). Além dos módulos presentes na Versão *Baseline*, a etapa FRC conta com o cálculo do tempo do grupo predito $T_{GP_{(i)}}$, do fator de correção $F_{C_{(i)}}$, do tempo do grupo predito ajustado $T_{GPA_{(i+1)}}$ e do *budget*. Já a etapa APOC possui o módulo de classificação e 10 tabelas de atualização distintas.

Conforme apresentado na Figura 33, a cada 16 quadros, o controlador gera o tempo desejado de codificação ajustado (T_{DA}), o qual é obtido a partir do tempo desejado (T_D) e do *budget* de tempo disponível. O controlador *CCAM* também obtém o tempo predito ajustado ($T_{GPA_{(i+1)}}$), com base no modelo de predição do tempo de codificação e do tempo de codificação observado no grupo de quadros anterior. A relação entre (T_{DA}) e ($T_{GPA_{(i+1)}}$), denominada $R_{TA_{(i+1)}}$, é usada para selecionar o próximo Ponto de Controle no conjunto de *PCs* apropriado, definido para o vídeo de entrada a

partir do modelo de especialização dos Pontos de Controle.

Mais detalhes sobre a implementação das melhorias/inovações apresentadas pelo controlador *CCAM* serão abordados nas seções seguintes. O pseudocódigo para este controlador pode ser observado no Apêndice E.

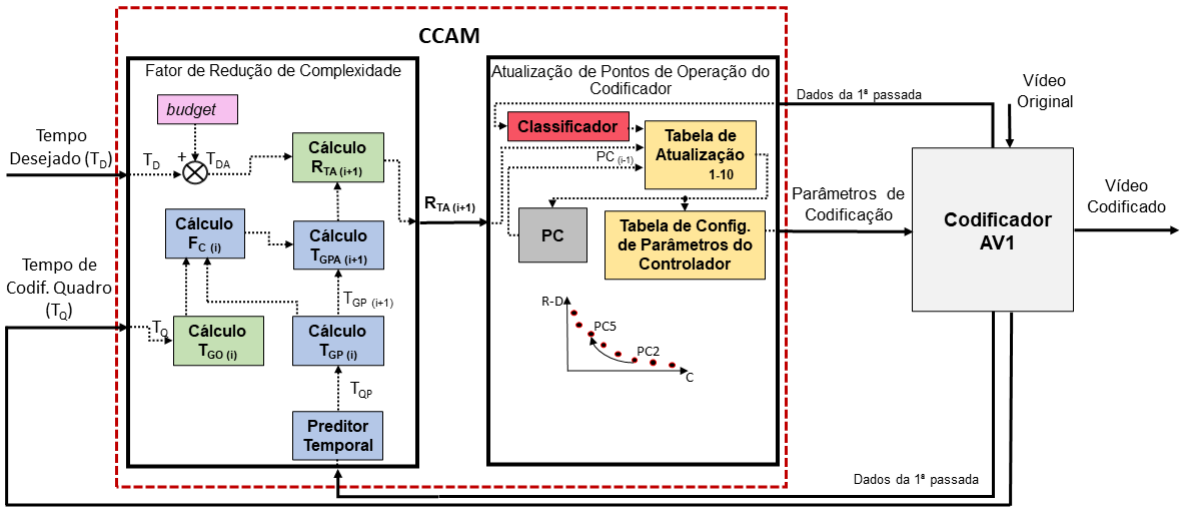


Figura 33 – Diagrama do controlador de complexidade adaptativo para o AV1 baseado em aprendizado de máquina.

6.4.1 Modelo de Predição do Tempo de Codificação

Diante da natureza dos dados, o modelo desenvolvido para a predição do tempo de codificação utiliza aprendizado supervisionado. Neste tipo de aprendizado de máquina, o *dataset* é formado por uma coleção de exemplos, cada um deles constituído por um vetor de atributos e um rótulo (BURKOV, 2019).

O *dataset* foi criado a partir dos primeiros 180 quadros de 70 sequências de vídeo recomendadas em (DAEDE; NORKIN; BRAILOVSKIY, 2020), (BOSSEN et al., 2013), (XIPH, 2022b) e (XIPH, 2022a)). Deste total, 34 sequências possuem resolução HD 1080 e 36 são de resolução UHD 4K. As sequências estão listadas no Apêndice G. Seis sequências de vídeo, selecionadas aleatoriamente, foram destinadas a etapa de teste. As sequências com resolução HD 1080 escolhidas foram: *BasketballDrive*, *in_to_tree* e *ParkScene*. Já para a resolução UHD 4K as sequências utilizadas foram: *Cosmos_aom_sdr_12916-13078*, *Meridian_aom_sdr_12264-12745* e *Nocturne_aom_sdr_27740-28109*. Foram usadas 64 sequências de vídeos para treino e validação e 6 para teste.

Inicialmente, as 42 informações, geradas na primeira passada de codificação Libaom, foram coletadas para serem utilizadas como atributos na construção do modelo. As mesmas podem ser observadas no Apêndice H. Também foram coletados os tempos de codificação de cada quadro para serem usados como rótulo ou valor alvo para os algoritmos de aprendizado de máquina. Cabe salientar que para obter esta

informação, o codificador precisou ser alterado, uma vez que o Libaom gera apenas o tempo total de codificação.

Apesar do tempo de codificação ser coletado individualmente por quadro, sua significativa variação entre os quadros vizinhos prejudicaria o treinamento do algoritmo de predição de tempo de codificação. Por isso, para cada quadro foi calculado o valor médio do tempo de codificação, como mostra a Figura 34. Analisando o comportamento promovido por diferentes quantidades de quadros, constatou-se que, na maioria das sequências, a utilização de 16 quadros torna a média aritmética mais estável. Logo, o valor médio do tempo de codificação de cada quadro foi obtido considerando 16 quadros, são eles: o quadro atual, os sete quadros posteriores e os oito quadros anteriores a ele, exceto para quadros localizados no início e fim da sequência. Nestes casos, a média foi calculada com os quadros disponíveis dentro dos limites da sequência.

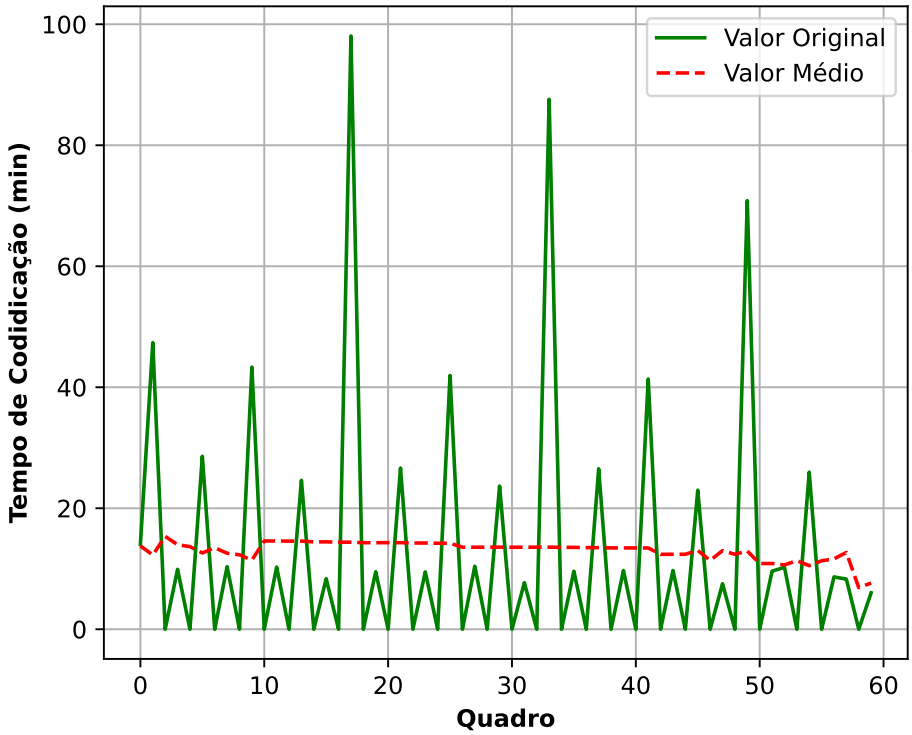
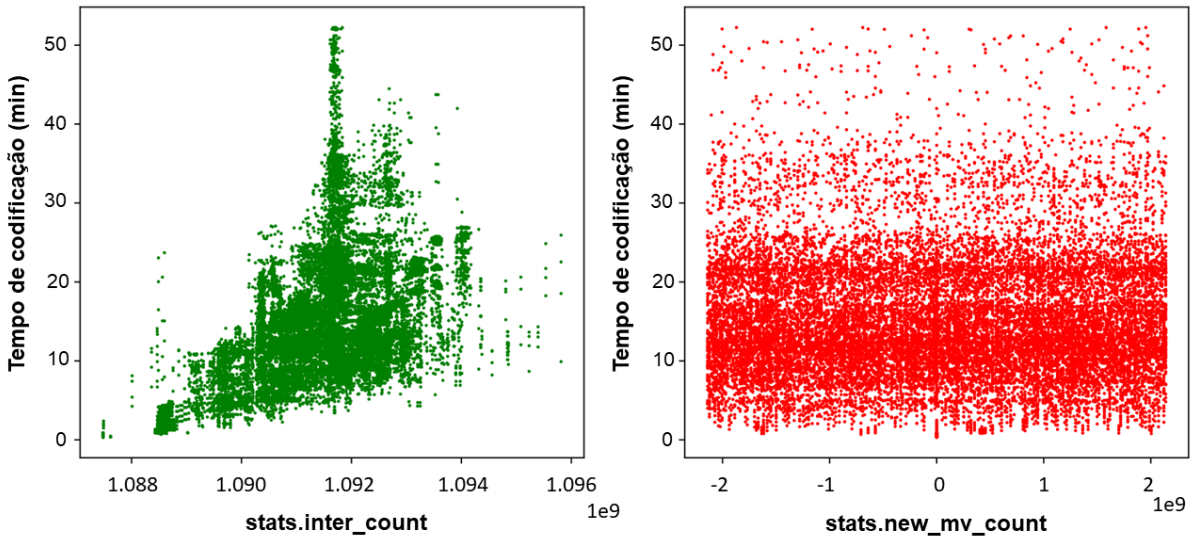


Figura 34 – Tempo de codificação de cada quadro e da média de 16 quadros da sequência *Netflix_SquareAndTimelapse_1920x1080*.

Posteriormente, a relação entre os atributos coletados e o tempo de codificação de cada quadro foi analisada. Para isso, a biblioteca Matplotlib (HUNTER, 2007) foi utilizada com a linguagem de programação Python. A Figura 35 exemplifica a análise realizada para dois atributos distintos: *stats.inter_count* (Figura 35 (a)) e *stats.new_mv_count* (Figura 35 (b)), onde cada ponto representa um quadro codifi-

cado.



(a) (b)

Figura 35 – Relação dos valores dos atributos stats.inter_counts (a) e stats.new_ms_count (b) com o tempo de codificação.

Na Figura 35 (a) é possível observar que há relação entre o tempo de cada quadro codificado e o valor do atributo em questão. Valores baixos deste atributo tendem a estar relacionados com tempos de codificação menores e vice-versa. Já na Figura 35 (b) é observado um comportamento aleatório entre o tempo de codificação do quadro e o atributo analisado, ou seja, pequenos valores deste atributo podem estar associados tanto a tempo de codificação menores quanto a tempo de codificação maiores.

Diante disso, 18 atributos que não apresentaram relação com o tempo de codificação foram descartados. Assim, além do valor de CQ usado em cada experimento, foram utilizados para o treinamento do modelo outros 25 atributos, apresentados no Apêndice H. Para organizar os arquivos gerados para o treinamento e facilitar a manipulação do *dataset* foi utilizada a biblioteca Pandas (MCKINNEY et al., 2011), com linguagem Phyton.

O modelo de predição de tempo de codificação foi treinado com algoritmos de aprendizado de máquina em Python, disponibilizados pelas bibliotecas *Scikit-Learn* (PEDREGOSA et al., 2011), XGBost (XGBOOST, 2022) e LGBM (MICROSOFT, 2022). Na biblioteca *Scikit-learn*, importantes ferramentas para acelerar o treinamento do modelo, como as relacionadas à divisão dos dados, validação cruzada e ajuste de hiperparâmetros da biblioteca, consideram que não haja dependência entre os exemplos existentes no *dataset*. Entretanto, a codificação de vídeo não se enquadra neste

cenário devido à redundância temporal existente. Os valores dos atributos e do tempo de codificação coletados entre os quadros vizinhos são, significativamente, relacionados.

Este fato caracteriza um "vazamento de dados" (SAMALA et al., 2020), pois o conjunto de treinamento e teste podem ser compostos por partes vizinhas da mesma sequência. Dessa forma, a similaridade entre os dados de treinamento e teste faria com que o modelo apresentasse uma "falsa" precisão, pois ao ser deparado com novos vídeos de entrada, a precisão, possivelmente, seria inferior a apresentada na fase de teste.

Este problema foi tratado com a utilização da validação cruzada com *k-folds* (HAN; KAMBER; J., 2011), aplicada no treinamento e na validação dos dados. Nela os dados foram separados por sequência de vídeo, garantindo que todos os quadros de uma determinada sequência estivessem presentes em apenas um conjunto: treinamento ou teste. Dessa forma, a possibilidade de vazamento de dados foi eliminada.

Para implementar este tipo de validação, foram gerados 10 grupos a partir de 64 sequências de vídeo. Nove desses grupos foram usados para treinamento e um para validação do modelo. Após 10 iterações de treino/validação com diferentes conjuntos de dados, os resultados de cada modelo foram definidos a partir da média do coeficiente de determinação (R^2), uma métrica de precisão popular para modelos de regressão (NASER; ALAVI, 2020), definida pela Equação 10. As variáveis A, $\bar{A}$ e P correspondem aos valores de rótulos, a média dos valores de rótulo e aos valores preditos, respectivamente.

$$\mathrm{R}^2 = 1 - \frac{\sum_{i=1}^{n} (P_i - A_i)^2}{\sum_{i=1}^{n} (A_i - \bar{A})^2} \tag{10}$$

A partir do método descrito anteriormente, os cinco modelos que apresentaram os melhores valores de métricas de precisão foram obtidos com os seguintes algoritmos de regressão: Regressor Linear, Ridge, Lasso, XGBoost e LBGM. Para estes modelos foi realizado o ajuste de hiperparâmetros (BURKOV, 2019), com exceção do Regressor Linear e LGBM, os quais apresentaram melhores resultados com as configurações padrão. Para cada hiperparâmetro, foram observados diversos valores e selecionado o que melhor impactou nas métricas de precisão. Para Ridge e Lasso foi usado o hiperparâmento de penalização com valor $4 * 10^6$. Já para o XGBoost seis hiperparâmetros foram modificados, são eles: objective='reg:squarederror', learning_rate=0.3, n_estimators=20, max_depth=7, min_child_weight=3, colsample_bytree=0.9. Os resultados da validação cruzada de cada modelo são apresentados na Tabela 19, assim como os resultados de cada iteração realizada.

A fim de otimizar os resultados, um único modelo de predição foi criado a partir

Tabela 19 – Resultados do treino de cada modelo selecionado

Iteração	**Linear** R^2	**LGBM** R^2	**Ridge** R^2	**Lasso** R^2	**XGBoost** R^2
1	0,14	0,27	0,40	0,27	-0,06
2	0,83	0,78	0,82	0,77	0,76
3	0,53	0,67	0,60	0,63	0,74
4	0,40	0,85	0,51	0,58	0,84
5	0,48	0,70	0,56	0,55	0,52
6	0,01	0,13	0,21	0,25	0,06
7	0,25	0,80	0,51	0,50	0,62
8	0,65	0,52	0,67	0,62	0,50
9	0,66	0,80	0,72	0,64	0,82
10	0,02	-0,55	0,19	-0,31	-0,48
Média	**0,40**	**0,50**	**0,52**	**0,45**	**0,43**

da combinação dos cinco modelos mencionados acima. Para isso, foi adotada uma das estratégias citadas por (BURKOV, 2019) para combinar diferentes modelos: a média. Como o nome sugere, nesta estratégia o valor médio das predições é calculado, considerando os resultados gerados por cada um dos modelos analisados. Assim, a validação cruzada com *k-folds* foi novamente aplicada. Desta vez, combinando os resultados dos modelos a partir do cálculo da média, com o uso da ferramenta *Voting Regressor*, da biblioteca SciKit-Learn. Os resultados para o modelo combinado são observados na Tabela 20. O valor médio de R^2 é 0,58, ou seja, a precisão melhorou em relação aos resultados obtidos individualmente por cada modelo (Tabela 19). Esta métrica indica o erro em relação ao tempo médio de codificação dos quadros, o qual para o *dataset* é de 30,77 minutos. Logo, o erro foi de, aproximadamente, 42%, o que corresponde a 13,58 minutos.

Tabela 20 – Resultados de validação cruzada do modelo combinado.

Iteração	**R^2**
1	0,54
2	0,83
3	0,68
4	0,72
5	0,61
6	0,18
7	0,69
8	0,69
9	0,82
10	0,04
Média	0,58

No conjunto de vídeos destinados ao teste do modelo, ocorreu uma pequena melhora no valor R^2, que foi de 0,60. Note que nesta métrica o valor ideal é 1, pois isto

significa que o modelo minimiza o erro em 100%, ou seja, o valor do erro é zero. Os valores de tempo de codificação reais e os valores preditos pelo modelo para cada quadro utilizado no teste são comparados no gráfico de dispersão apresentado na Figura 36. Como pode ser observado, o erro é mais expressivo para tempos de codificação reais maiores que 60 minutos. Nesses casos os pontos estão mais distantes da linha vermelha, a qual representa erro zero.

Este comportamento foi influenciado por duas sequências: *Cosmos_aom_12916-13078* e *Nocturne_aom_sdr_27740-28109*. A primeira não segue o padrão da grande maioria das sequências usadas para o treinamento, no qual o tempo de codificação decresce conforme o valor do CQ aumenta. Já a segunda apresenta, para os CQs 20 e 32, tempos de codificação próximos a 100 e 80 minutos por quadro, respectivamente, os quais podem ser considerados elevados se comparados a sequências de vídeo usadas para o treinamento. Assim, o modelo não conseguiu prever tais comportamentos e gerou valores que subestimam o tempo de codificação. Mais detalhes sobre os resultados das sequências *Cosmos_aom_12916-13078* e *Nocturne_aom_sdr_27740-28109* podem ser observadas, respectivamente, nas Figuras 67 e 69 no Apêndice I, assim como os resultados dos demais vídeos usados para testar o modelo final.

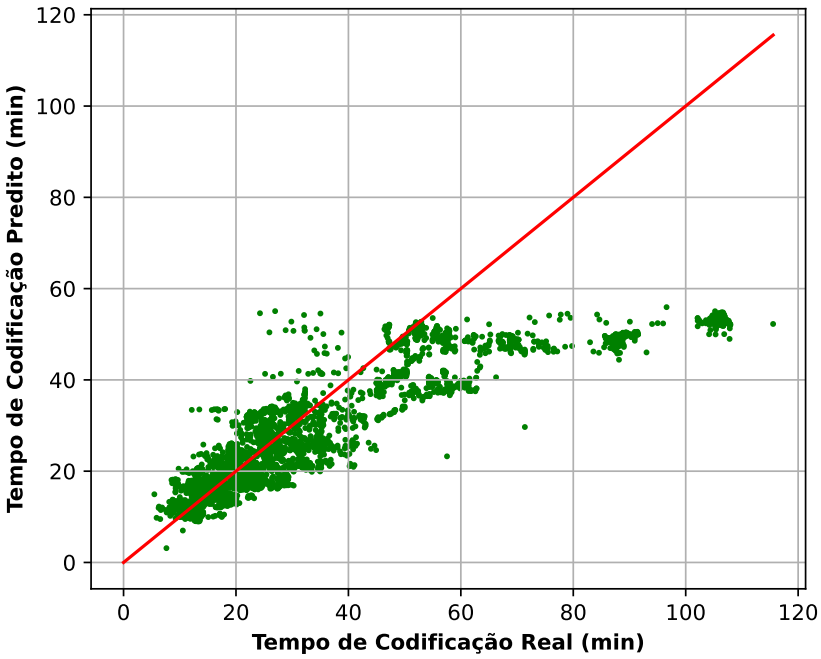


Figura 36 – Relação dos valores preditos e valores reais obtidos.

O modelo, apesar de apresentar erro de predição relativamente distante do ideal, é capaz de representar adequadamente as alterações sofridas no tempo de codificação da sequência de entrada. No contexto do controlador adaptativo de complexidade para o AV1, isto pode ser interpretado como uma vantagem em relação a Versão *Baseline*,

pois está mais próximo de retratar as variações no tempo de codificação entre os grupos de quadros do vídeo que está sendo codificado.

6.4.2 Modelo para Especialização dos Pontos de Controle

Para especializar o controlador de acordo com características do vídeo de entrada foram desenvolvidos modelos de clusterização, que são classificados como modelos de aprendizado de máquina não supervisionados (BURKOV, 2019), isto é, não existem rótulos ou valores “verdade” para todos os exemplos que serão preditos. Utilizando o algoritmo K-Means (BURKOV, 2019) da biblioteca SciKit-Learn, linguagem Python, foi criado um modelo para resolução HD 1080 e outro para resolução UHD 4K.

O algoritmo k-Means é um processo iterativo de mover os centros dos *clusters*, denominados centroides, à posição média de seus pontos constituintes e reatribuir instâncias (quadros de codificação) a seus *clusters* mais próximos, iterativamente, até que não haja mudança significativa no número de centroides possíveis ou número de iterações alcançado (DANGETI, 2017). A Figura 37 ilustra a ideia de clusterização, onde o conjunto de dados da Figura 37 (a) são atribuídos aos quatro *clusters* previamente definidos e presentes na Figura 37 (b), representados com cores distintas.

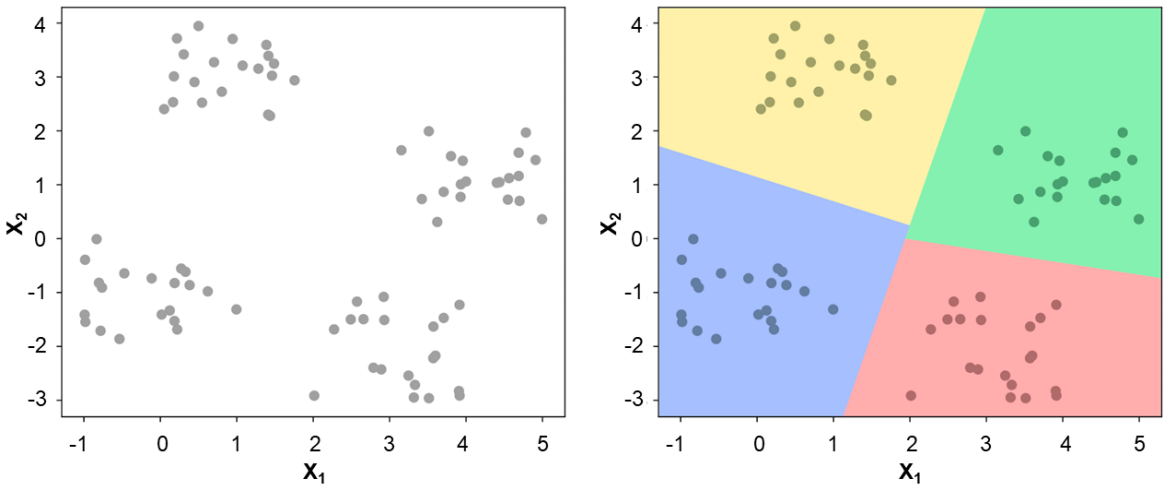


Figura 37 – Conjunto de dados (a) e agrupamento dos dados em quatro *clusters* (b).

Os *datasets* foram criados a partir dos dados coletados na primeira passada Libaom das cinco sequências de vídeo com resolução HD 1080 e cinco sequências UHD 4K usadas para análise de sensibilidade, apresentadas no Capítulo 5. A lista das informações usadas como atributos pela etapa de treinamento encontra-se no Apêndice H. Com base nessas informações, foram criados cinco *clusters* para cada um dos modelos. Após a distância Euclidiana (DANGETI, 2017) em relação a centroide ser calculada, para cada quadro do vídeo de entrada, o quadro é atribuído ao *cluster* com centroide mais próximo.

O fato dos modelos de clusterização serem não-supervisionados impossibilita a verificação da precisão dos mesmos. Por isso, as classes selecionadas a partir do modelo proposto foram comparadas com as obtidas a partir das informações SI e TI do vídeo de entrada em relação aos vídeos usados na análise de complexidade, Capítulo 5 (mais detalhes são apresentados no Apêndice J). Os experimentos foram realizados com os seguintes vídeos HD 1080: *pan_hdr_amazon_1080p*, *seaplane_hdr_amazon_1080p* e *station2_1080p25_60f*. Foram usados 180 quadros de cada sequência e os quatro CQs recomendados em (DAEDE; NORKIN; BRAILOVSKIY, 2020).

As classes selecionadas para as sequências *pan_hdr_amazon_1080p*, *seaplane_hdr_amazon_1080p* e *station2_1080p25_60f*2, com o uso das informações SI e TI, foram, respectivamente: "guitar", "square" e "crosswalk". Nas sequências *seaplane_hdr_amazon_1080p* e *station2_1080p25_60f*, as classes selecionadas com o modelo desenvolvido, vão ao encontro das classes obtidas com as informações SI e TI. O modelo proposto utiliza a classe "square" em 98,98% dos quadros da sequência *seaplane_hdr_amazon_1080p* e usa a classe "crosswalk" em 65,56% dos quadros da sequência *station2_1080p25_60f*. Já na sequência *pan_hdr_amazon_1080p*, o modelo proposto utiliza em 55,56% dos quadros a classe "crosswalk", a qual é a segunda classe mais indicada para esta sequência segundo as informações espaciais e temporais. Os resultados apontam que as classes não são selecionadas aleatoriamente pelo modelo desenvolvido, demonstrando coerência no comportamento do mesmo.

6.4.3 Implementação dos Modelos de Aprendizado de máquina no Controlador

Para viabilizar a implementação dos modelos no codificador AV1, foi necessária a conversão para a linguagem de programação usada no Libaom (linguagem C), com o auxílio da biblioteca para Python M2CGen (GITHUB, 2022). Também foi desenvolvida uma função específica para coletar os dados da primeira passada durante o processo de codificação. Assim, as informações, obtidas a cada quadro, são organizadas em vetores e enviadas para os modelos desenvolvidos. Os resultados dos modelos, gerados quadro a quadro, são armazenados e usados pelo controlador de complexidade na segunda passada de codificação.

6.4.3.1 Implementação do Modelo de Predição de Tempo

Durante a codificação, podem ser observadas discrepâncias entre os tempos dos quadros, que ocorrem, principalmente, nas trocas de cena e no primeiro quadro da sequência. Esses casos foram tratados com base na distribuição normal de um grupo de valores (LAMORTE, 2022), na qual 68% dos valores estão posicionados a um desvio padrão do valor médio, 95% para duas vezes o desvio padrão e 99% para três desvios padrão, como mostra a Figura 38. Nela σ representa o desvio padrão e μ está

relacionado a média dos valores.

Dessa forma, valores preditos com diferença maior que três desvios padrão em relação ao valor médio predito foram considerados discrepantes. Estes valores foram substituídos pelo valor médio do tempo predito da sequência em questão. Na Figura 39 é possível observar esta estratégia sendo aplicada na sequência *In_to_tree* com CQ 43. Note que o valor discrepante, o qual encontra-se circulado no primeiro gráfico, é removido enquanto os outros valores se mantêm inalterados.

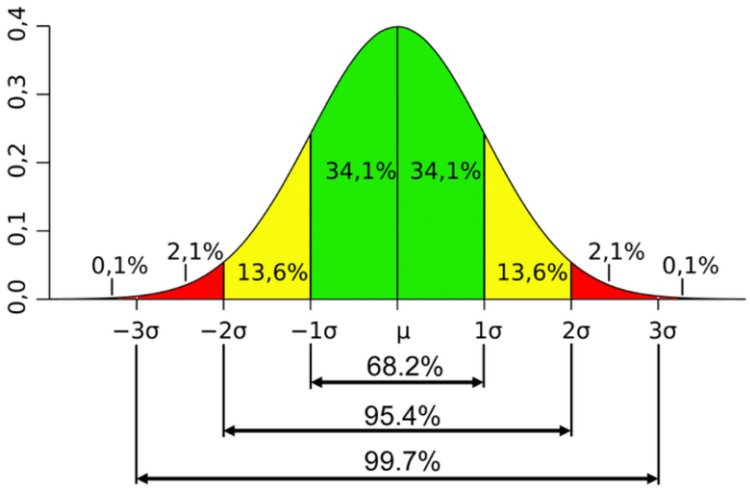
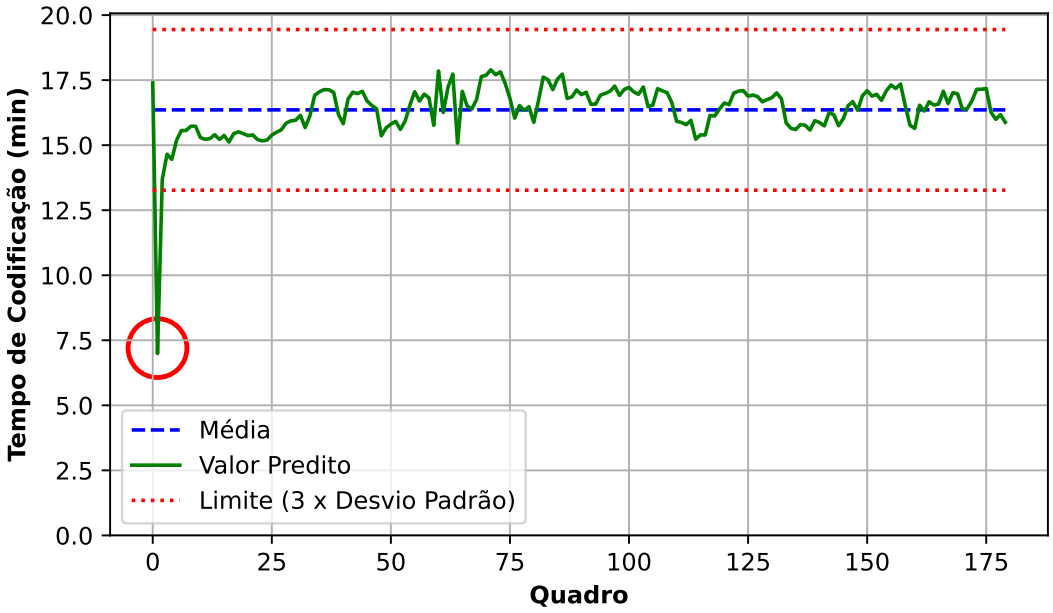


Figura 38 – Relação de uma distribuição normal e desvio padrão. Fonte: Lamorte (2022).

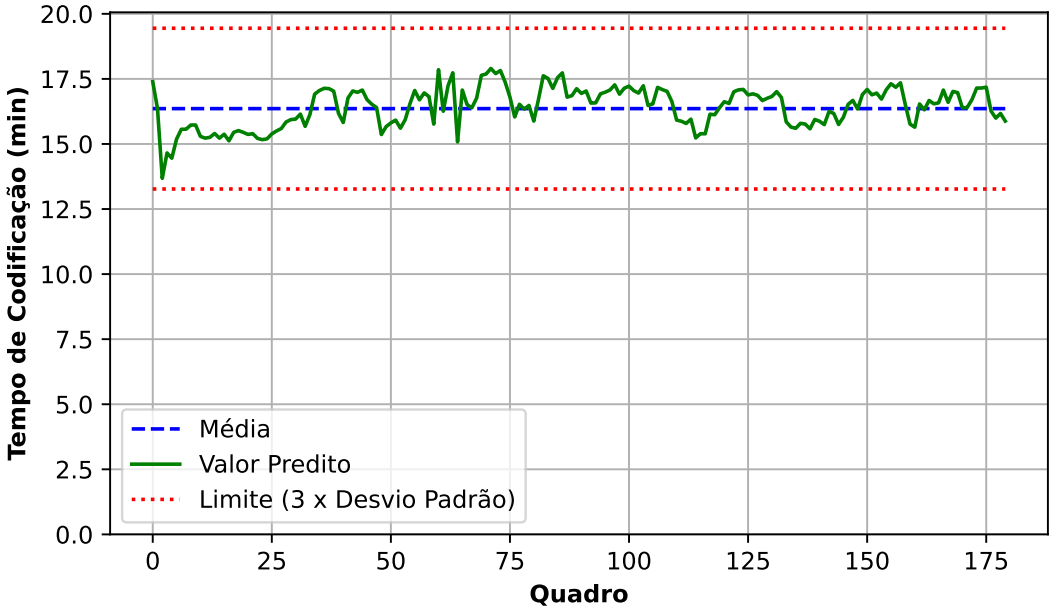
O modelo de predição de tempo de codificação (Seção 6.4.1) gera resultados para cada quadro, porém, o passo de controle é aplicado a cada grupo de 16 quadros. Logo, para implementar este modelo no controlador é necessário que se obtenha o tempo de codificação predito para cada grupo $T_{GP_{(i)}}$. Conforme a Equação 11, o tempo de codificação de cada grupo é obtido pelo somatório dos tempo de codificação predito, $T_{QP_{(x)}}$, dos 16 quadros que compõe o respectivo grupo.

$$T_{GP_{(i)}} = \sum_{x=16(i-1)+1}^{16i} T_{QP_{(x)}} \tag{11}$$

Além disso, o tempo predito do próximo grupo a ser codificado é ajustado, $T_{GPA_{(i+1)}}$, a partir da multiplicação por um fator de correção ($F_{C_{(i)}}$), como mostra a Equação 12. O fator de correção considera a relação entre o tempo observado, $T_{GO_{(i)}}$, (Equação 7) e o tempo predito do último grupo codificado, $T_{GP_{(i)}}$, conforme pode ser observado na Equação 13. Como o cálculo do fator de correção depende de informações do grupo de quadros anterior, ele só é aplicado a partir do grupo 2. A Figura 40 exemplifica a aplicação do fator de correção entre os grupos de quadros 2 e 5, para a sequência *rush_hour_1080p25_60f*. Note que o tempo predito de cada grupo se torna mais próximo do tempo de codificação real em todos os casos.



(a)



(b)

Figura 39 – Remoção de valores discrepantes, usando desvio padrão. Sequência *In_to_tree* com CQ 43.

$$T_{GPA_{(i+1)}} = T_{GP_{(i+1)}}.F_{C_{(i)}} \tag{12}$$

$$F_{C_{(i)}} = \frac{T_{GO_{(i)}}}{T_{GP_{(i)}}} \tag{13}$$

O tempo desejado ajustado, T_{DA} (Equação 14) e o *budget* (Equação15) são usados para calcular o valor do sinal R_{TA}, Equação 16, o qual será utilizado pelo controlador de complexidade para a selecionar o *PC* do próximo grupo de quadros a ser codificado.

$$T_{DA} = T_D + \textit{budget} \tag{14}$$

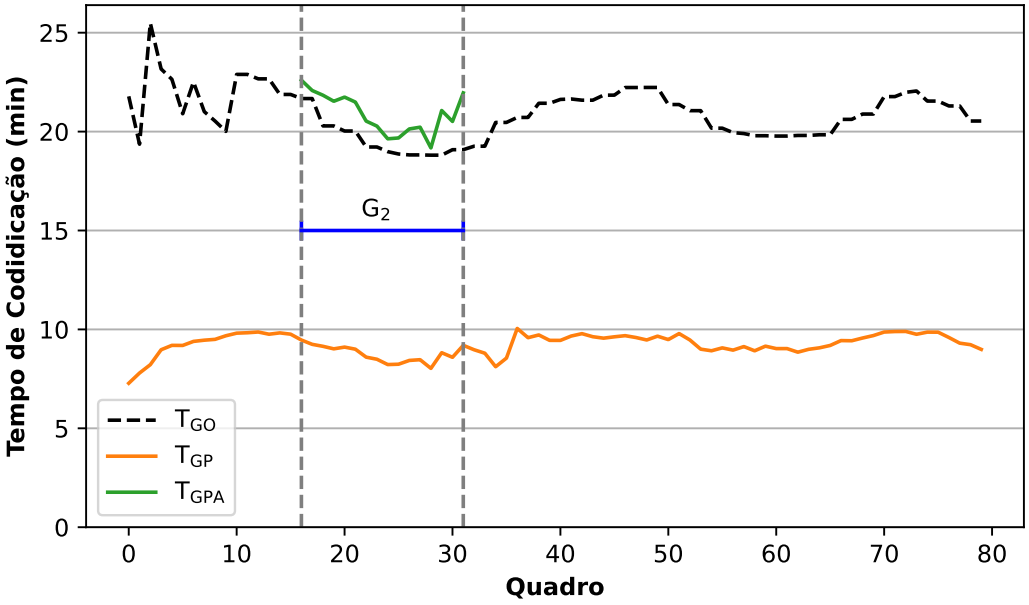
$$\textit{budget} = \sum_{i-1}^{j=1} T_{D(j)} - T_{GO(j)} \tag{15}$$

$$R_{TA_{(i+1)}} = \frac{T_{DA}}{T_{GPA_{(i+1)}}} \tag{16}$$

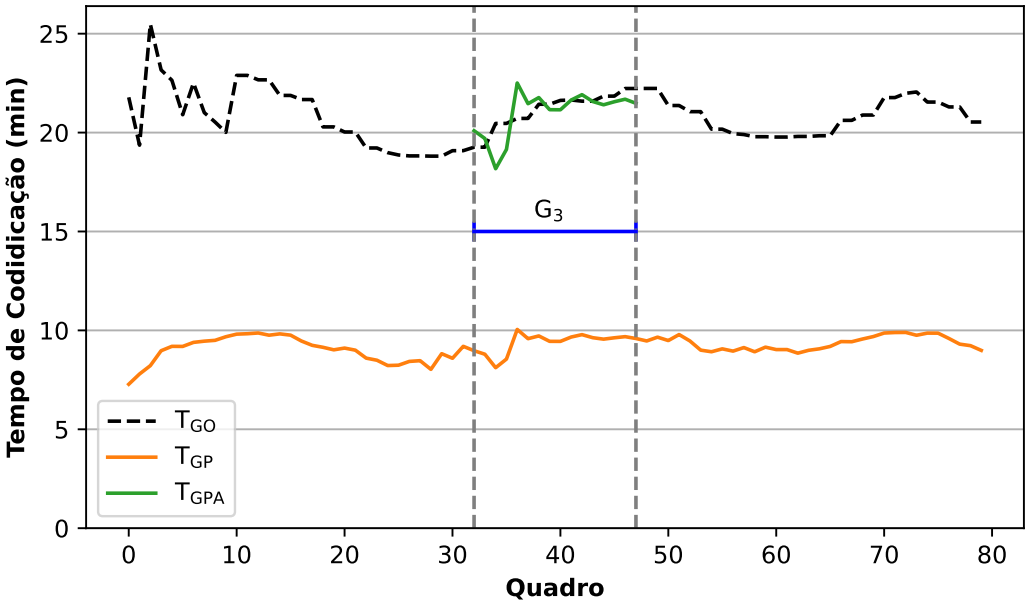
6.4.3.2 Implementação do Modelo para Especialização do Controlador

Na primeira passada de codificação, os dados são extraídos e enviados para os modelos, quadro a quadro. Um modelo, então, é selecionado de acordo com a resolução do vídeo de entrada. Assim, cada quadro do vídeo a ser codificado é atribuído para um dos cinco *clusters* existentes e os resultados são armazenados. Para viabilizar a implementação do modelo no controlador é necessário que, na segunda passada, a classificação seja implementada a cada grupo de 16 quadros. Diversos modos de implementação foram avaliados, como mostra, em maiores detalhes, o Apêndice J. Entre eles, diante dos resultados apresentados, o modo "número de ocorrência ponderada" foi o escolhido para ser utilizado no controlador.

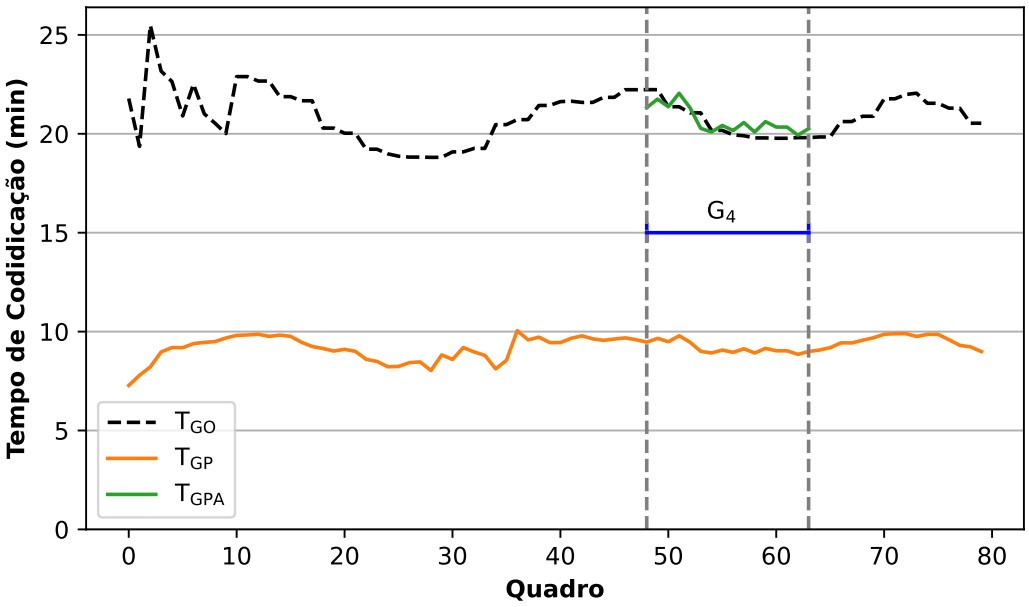
Neste modo, a ponderação é realizada considerando a estrutura e o índice dos quadros no GFG. Os pesos atribuídos aos resultados de cada quadro foram definidos com base no comportamento do tempo dos quadros no GFG de tamanho 16, apresentado na Figura 31. Quanto maior o tempo de codificação do quadro, maior é o peso relacionado a ele e vice-versa. Porém, o GFG pode ser composto por menos de 16 quadros. Nestes casos, é necessário que ocorra o alinhamento dos pesos. Isto é possível a partir da variável *frames_till_gf_udate_due*, a qual indica quando ocorrerá a atualização do GFG. Na Figura 41 as barras em verde mostram os pesos atribuídos a cada quadro, considerando que o GFG seja composto por 16 quadros, já as barras laranja apresentam o alinhamento dos pesos considerando que, o GFG será atualizado no quinto quadro. Note que os pesos foram deslocados cinco quadros, para alinhar


Tempo de Codidicação (min)
Quadro
G2
TGO
TGP
TGPA

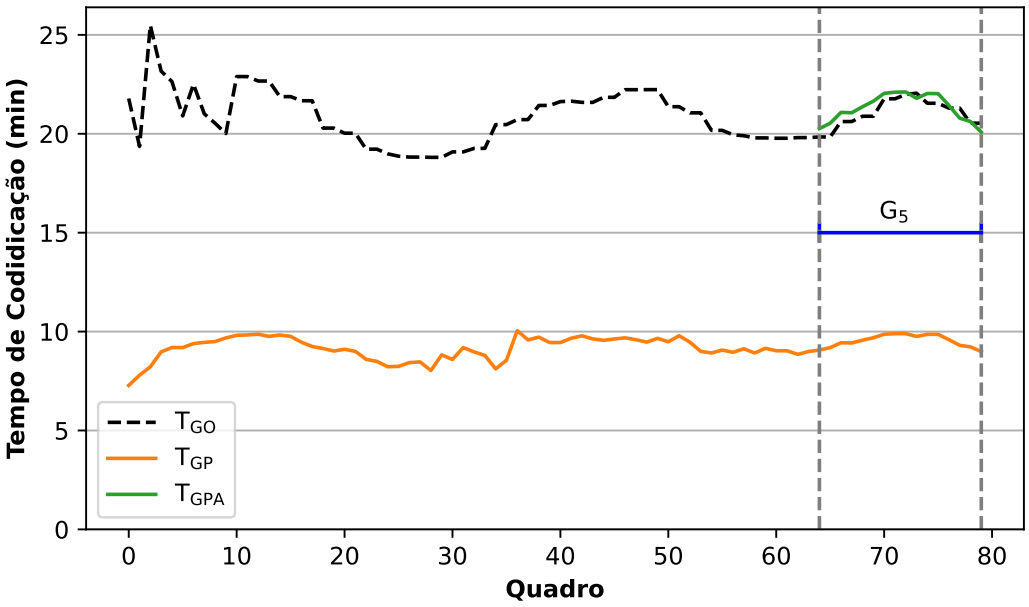
(a)


Tempo de Codidicação (min)
Quadro
G3
TGO
TGP
TGPA

(b)



(c)



(d)

Figura 40 – Fator de correção aplicado dos grupos 2 ao 5 da sequência *rush_hour_1080p25_60f*

com a atualização do GFG.

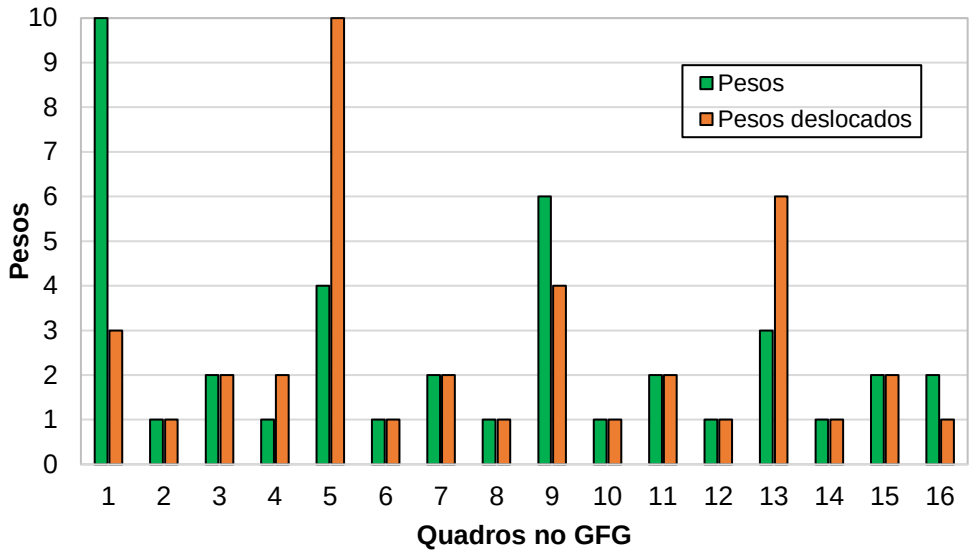


Figura 41 – Distribuição dos pesos de cada quadro.

Após aplicada a ponderação, a classe mais presente entre os 16 quadros que constituem o grupo é escolhida para codificá-lo. Com esta informação, o controlador busca o próximo PC na tabela de transição correspondente a classe selecionada. Para isso, os 10 conjuntos de *PCs* referentes a cada um dos vídeos usados na análise de complexidade, Capítulo 5, foram implementados no controlador. Os Pontos de Controle referentes a cada uma dessas 10 sequências são apresentados no Apêndice D, já suas tabelas de transição são apresentadas no Apêndice F.

7 RESULTADOS

Este capítulo apresenta o *setup* e a metodologia de teste empregada na geração dos resultados finais dos testes objetivos realizados para os controladores desenvolvidos, assim como os resultados de precisão no controle de redução de complexidade e de eficiência de codificação obtidos para o controlador de complexidade Versão *Baseline* (Seção 6.3) e o controlador de complexidade baseado em aprendizado de máquina, denominado *CCAM*, (Seção 6.4) quando aplicados na codificação de vídeos de resolução HD e UHD. Além disso, este capítulo também apresenta os resultados referentes aos testes de avaliação de qualidade subjetiva, realizados para o controlador *CCAM* na codificação de vídeos de resolução HD 1080.

7.1 *Setup* de Experimentos e Metodologia para os Testes Objetivos

Os experimentos foram realizados em um computador com processador Intel Xeon E5-4650v3 com noventa e seis núcleos de 2,10 GHz e memória de 512GB, utilizando o software de referência do AV1 Libaom 2.0.0 (*hash* 72824a7) (AOMEDIA, 2018a). Para os testes com resolução UHD 4K foram usadas sequências de vídeo recomendadas em (XIPH, 2022a), devido ao limitado número de sequências de vídeos UHD 4K disponíveis em (DAEDE; NORKIN; BRAILOVSKIY, 2020).

Novamente, a seleção das sequências de vídeo foi baseada em uma análise do índice de atividade espacial e do índice de atividade temporal. Dessa forma, um conjunto heterogêneo de sequências foi analisado para gerar as conclusões das avaliações referentes aos controladores desenvolvidos. Na Figura 42 é possível observar a relação SI×TI (eixos X e Y, respectivamente) das 21 sequências de vídeo HD 1080 definidas pelo IETFNETVC-Testing. As cinco sequências com resolução HD 1080 utilizadas estão destacadas (círculos vermelhos) na Figura 42, são elas: *rush_hour_1080p25_60f*, *Netflix_FoodMarket_1920x1080_60fps_8bit_420_60f*, *Netflix_PierSeaside_1920x1080_60fps_8bit_420_60f*, *Netflix_Boat_1920x1080_60fps_8bit_420_60f* e *park_joy_1080p50_60f*. A Figura

43 ilustra a relação do índice de atividade espacial e do índice de atividade temporal das 50 sequências de vídeo UHD 4K, disponíveis em (DAEDE; NORKIN; BRAILOVSKIY, 2020) e (XIPH, 2022a). As cinco sequências de vídeos UHD 4K utilizadas encontram-se destacadas por círculos vermelhos na Figura 43: *Netflix_Dancers_4096x2160_60fps_10bit_420_60f*, *Netflix_WindaAndNature_4096x2160_60fps_10bit_420_60f*, *sol_levante_aom_sdr_519-649_180f*, *sparks_aom_sdr_6026-6502_180f* e *FlowerSky_A*.

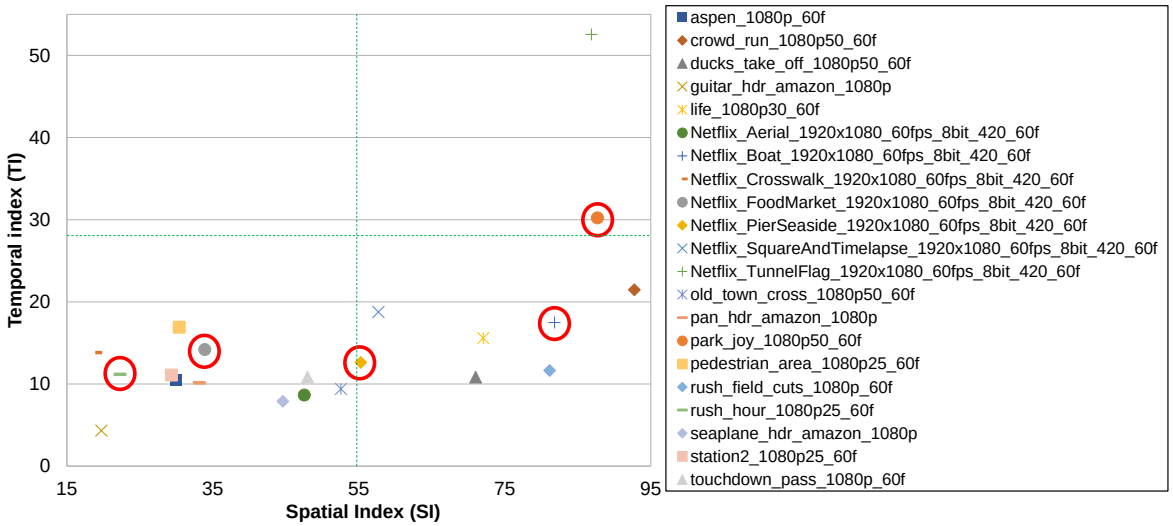


Figura 42 – Índice de atividade espacial e temporal para sequências de vídeos com resolução HD 1080 recomendadas em (DAEDE; NORKIN; BRAILOVSKIY, 2020).

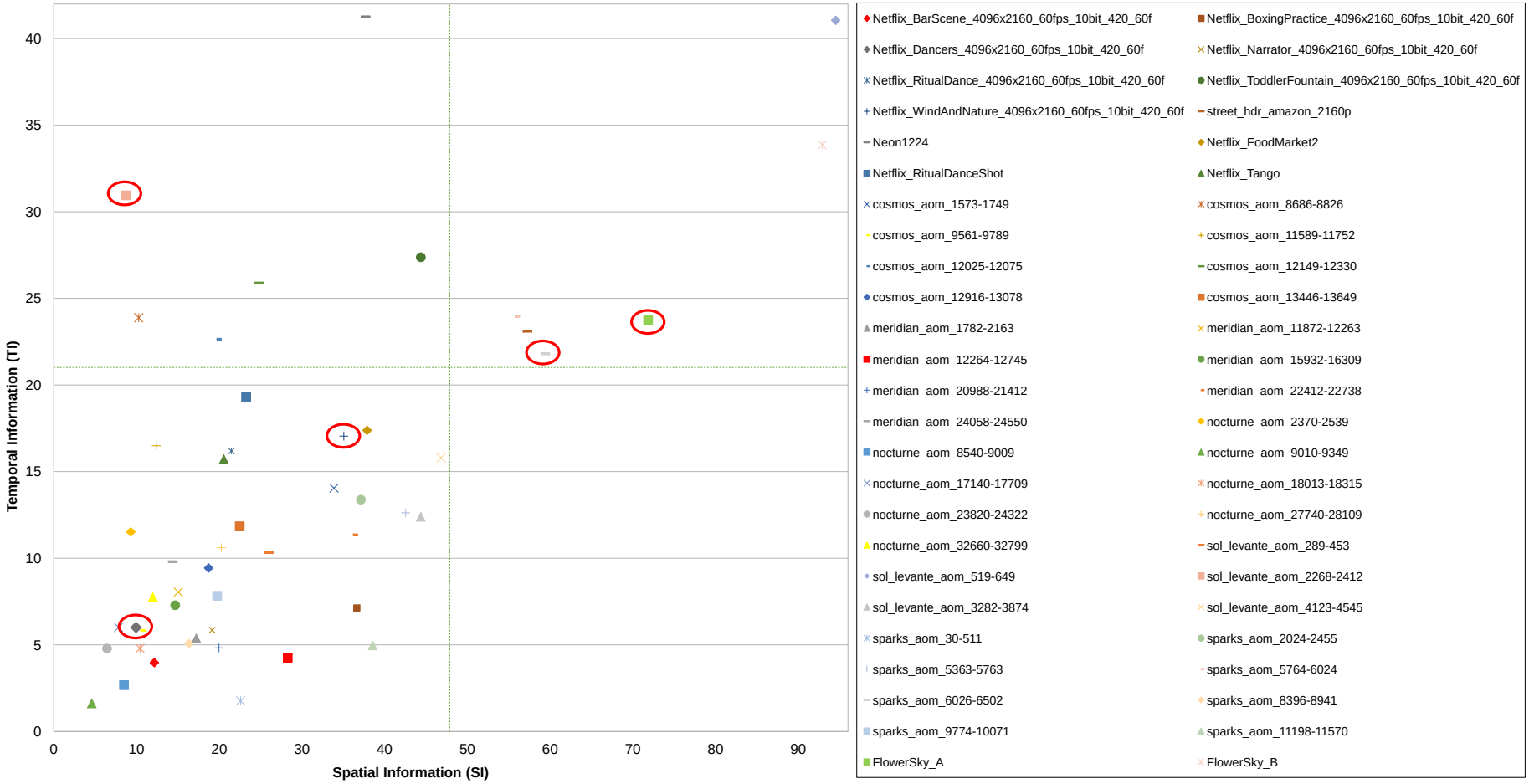


Figura 43 – Índice de atividade espacial e temporal para sequências de vídeos com resolução HD 4K recomendadas em (DAEDE; NORKIN; BRAILOVSKIY, 2020) e (XIPH, 2022a).

Foram analisados os 180 quadros de cada sequência de teste para os quatro CQs (20, 32, 43 e 55) recomendados em (DAEDE; NORKIN; BRAILOVSKIY, 2020). Os controladores foram avaliados com sete valores diferentes de redução de tempo de codificação alvo, são eles: 10%, 20%, 30%, 40%, 50%, 60% e 70%. Para cada sequência de teste avaliada foram coletados os seguintes resultados: tempo de codificação com a configuração padrão do Libaom (T_{CP}), tempo observado de cada grupo de quadros ($T_{GO_{(i)}}$), tempo desejado (T_D), tempo desejado ajustado (T_{DA}), Pontos de Controle selecionados, sinais $R_{T_{(i+1)}}$ e $R_{TA_{(i+1)}}$, *budget*, classe do vídeo de entrada, PSNR e *bitrate*.

Um teste mais longo também foi realizado, considerando a sequência de vídeo *speed_bag*, disponibilizada em (XIPH, 2022b). Esta sequência, com resolução HD 1080, possui uma quantidade de quadros mais expressiva, totalizando 570 quadros. Para a sequência *speed_bag*, além dos sete alvos de redução de tempo citados anteriormente, também foi realizado um experimento no qual o tempo desejado foi alterado durante a codificação. Neste experimento, nos primeiros 18 grupos de quadros (289 quadros) a meta estipulada foi de 10% de redução de tempo de codificação e, no restante dos grupos (281 quadros), a redução de tempo desejada foi ajustada para 30%.

Além disso, também foi analisado a sequência de vídeo *speed_bag*, disponibilizado em (XIPH, 2022b). Esta sequência, com resolução HD 1080, possui uma quantidade de quadros mais expressiva, 570 quadros. Os tempos desejados foram analisados com sete valores de redução de tempo de codificação alvo, são elas: 10%, 20%, 30%, 40%, 50%, 60% e 70%. Para a sequência *speed_bag*, além das reduções de tempo citadas acima, foi realizado um experimento no qual o tempo desejado foi alterado durante a codificação. Neste experimento, nos primeiros 18 grupos de quadros a meta estipulada foi de 10% de redução de tempo de codificação e no restante dos grupos a redução de tempo desejada foi de 30%.

7.2 Resultados Objetivos dos Controladores Desenvolvidos

Esta seção apresenta diversos resultados objetivos, importantes para avaliar o desempenho dos controladores propostos. Para ambos controladores, Versão *Baseline* e *CCAM*, são apresentados resultados obtidos a partir da análise do comportamento dinâmico, do CQ e dos valores médios. Além disso, são apresentados os resultados obtidos com a variação da RTC alvo em tempo de execução e uma comparação entre o controlador *CCAM* e outros controladores de complexidade.

7.2.1 Análise do comportamento dinâmico dos controladores

Para analisar o comportamento dinâmico dos controladores desenvolvidos nesta tese, são usados, como exemplo, os resultados obtidos para RTC de 50% com a sequência de vídeo *Netflix_Boat_1920x1080_60fps_8bit_420_60f*, CQ 32. Na Tabela 21 é possível observar os resultados obtidos para RTC de 50% com a sequência de vídeo *Netflix_Boat_1920x1080_60fps_8bit_420_60f*, CQ 32, usando a Versão *Baseline* do controlador. Para cada grupo de 16 quadros, são apresentados: o tempo de codificação com a configuração padrão (T_{CP}) (sem compressão), o tempo de codificação desejado (T_D), o tempo de codificação observado ($T_{GO_{(i)}}$), o sinal $R_{T_{(i+1)}}$, o *PC* selecionado para o próximo grupo de quadros e a RTC obtida. Além disso, também pode ser observada a RTC média, considerando todos os grupos analisados. Para cada experimento, o tempo desejado de codificação (T_D) foi determinado pelo usuário, de maneira *off-line*, conforme a Equação 8, e o primeiro grupo de quadros foi usado como âncora.

A Versão *Baseline* do controlador seleciona os Pontos de Controle de acordo com $R_{T_{(i+1)}}$, que é um sinal de controle gerado a partir do tempo de codificação desejado (T_D) e do tempo de codificação observado em cada grupo de quadros ($T_{GO_{(i)}}$). Todos os processos iniciam usando *PC0*. Tendo os resultados da Tabela 21 como exemplo, após obter a relação entre o tempo desejado e o tempo observado de cada grupo (Equação 9), é gerado o sinal de controle com valor de 0,62. Na Tabela 16, na coluna do *PC* anterior (*PC0*) o algoritmo busca o menor valor mais próximo a 0,62 que, neste caso, é 0,56. Então, o grupo 2 é codificado com o *PC20*. A partir dos resultados obtidos para o grupo 2 é gerado um novo sinal de controle, com valor 0,89, e o mesmo processo é repetido, onde, desta vez, o *PC24* é selecionado. O grupo 3 é então codificado com o *PC24*. Este processo segue sendo aplicado até a conclusão da codificação. Dessa forma, a Versão *Baseline* do controlador responde corretamente aos estímulos, selecionando as configurações de acordo com os valores informados pelas entradas. Porém, a diferença entre a percentagem do tempo de codificação desejada e a obtida é relativamente alta, atingindo 8,25 pontos percentuais.

Os resultados podem ser observados, graficamente, nas Figuras 44, 45 e 46. Na Figura 44, as linhas azul, cinza, laranja, amarelo e verde representam, respectivamente, o tempo de codificação obtido a partir da configuração padrão (T_{CP}), a média do tempo de codificação padrão considerando todos os grupos de quadros, o tempo de codificação desejado (T_D), o tempo observado para cada grupo de quadros ($T_{GO_{(i)}}$) e a média do tempo observado considerando os 11 grupos de quadros analisados. Os resultados da Figura 44 mostram que no grupo 1 é onde o T_{CP} se distancia mais do valor médio do tempo de codificação com a configuração padrão. Além disso, é possível observar que até o terceiro grupo de quadros, o tempo observado de cada grupo é sempre maior que o tempo de codificação desejado. A partir do grupo de

quadros 4, o tempo de codificação dos grupos assume tanto valores menores quanto valores maiores que o tempo desejado. Em média, o tempo observado de cada grupo é, aproximadamente, 10 minutos menor que o tempo desejado.

Tabela 21 – Resultados obtidos com a Versão *Baseline* do controlador para o vídeo Netflix_Boat_1920x1080_60fps_8bit_420_60f, CQ 32, considerando RTC alvo de 50%.

Grupo	T_{CP} (min)	T_D (min)	T_{GO} (min)	R_T	Próximo PC	RTC (%)	Média RTC (%)
1	171,83	105,68	170,81	0,62	20	0,59	
2	224,66	105,68	118,50	0,89	24	47,25	
3	227,08	105,68	133,36	0,79	26	41,27	
4	51,93	105,68	51,93	2,04	24	73,58	
5	199,03	105,68	111,87	0,94	26	43,79	
6	204,24	105,68	52,16	2,03	24	74,46	58,25
7	216,43	105,68	130,81	0,81	26	39,56	
8	192,70	105,68	51,21	2,06	24	73,42	
9	213,71	105,68	123,16	0,86	26	42,37	
10	217,22	105,68	55,15	1,92	26	74,61	
11	224,32	105,68	55,31	1,91	26	75,35	

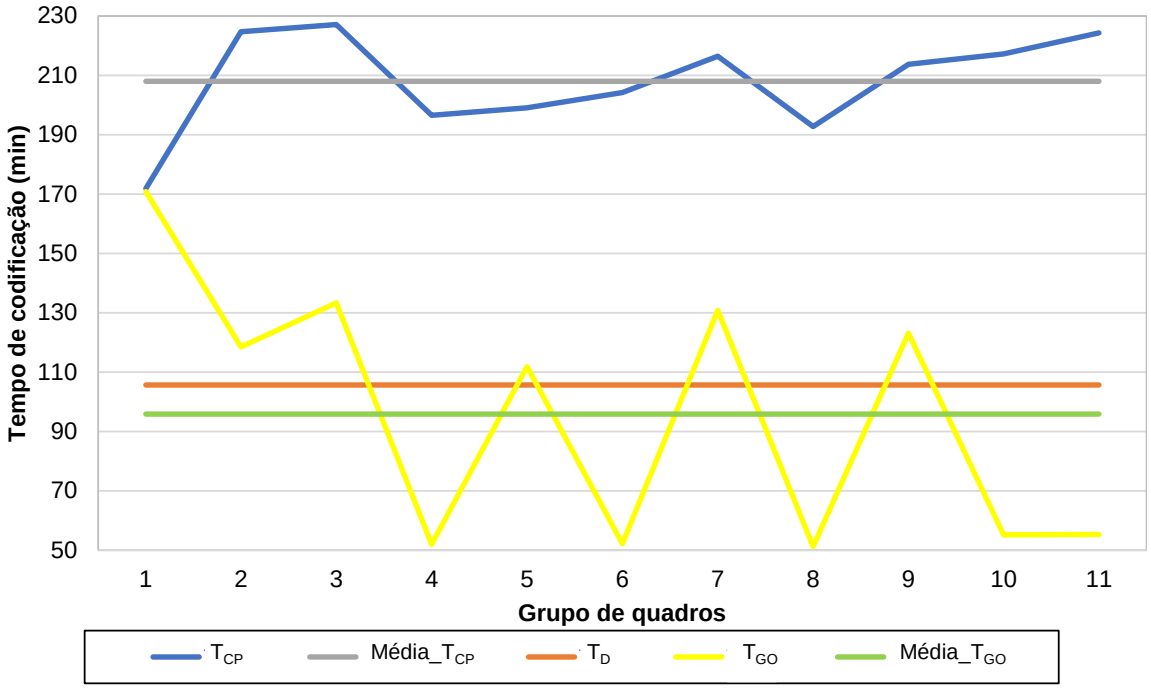


Figura 44 – Resultados obtidos com a Versão *Baseline* do controlador para a sequência de vídeo Netflix_Boat_1920x1080_60fps_8bit_420_60f, CQ 32, para a RTC alvo de 50%.

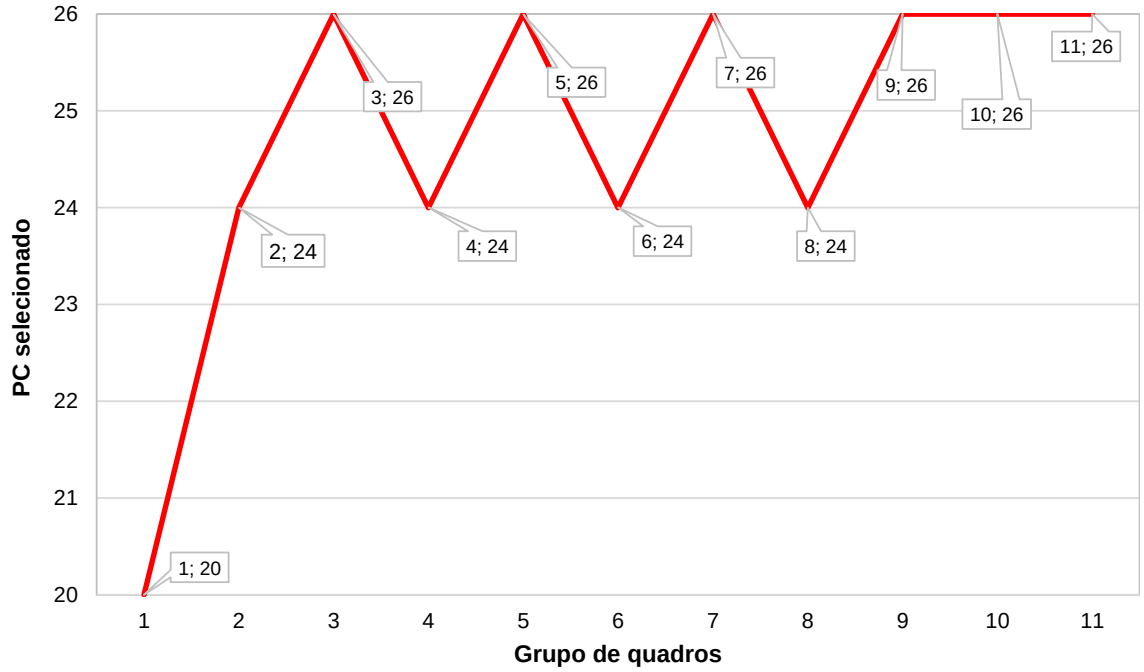


Figura 45 – Próximo Ponto de Controle selecionado, obtido com a Versão *Baseline* do controlador, na sequência de vídeo Netflix_Boat_1920x1080_60fps_8bit_420_60f, CQ 32, para a RTC alvo de 50%.

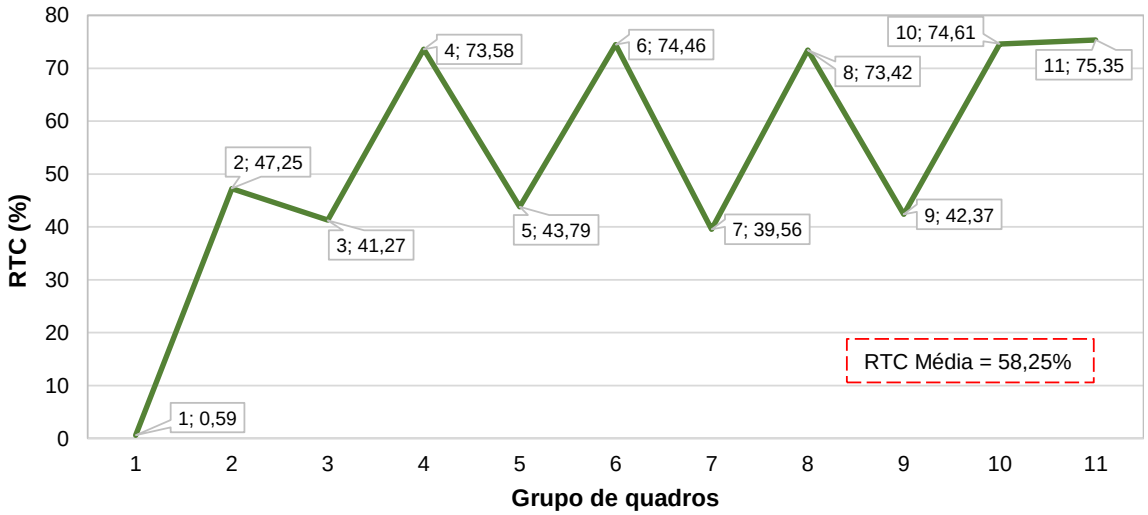


Figura 46 – Redução de tempo de codificação, obtida com a Versão *Baseline* do controlador, para a sequência de vídeo Netflix_Boat_1920x1080_60fps_8bit_420_60f, CQ 32, para a RTC alvo de 50%.

A Figura 45 apresenta o *PC* selecionado em cada grupo de quadros, o qual é aplicado na codificação do grupo de quadros seguinte. Os valores dos rótulos representam "grupo; próximo PC". De acordo com a Figura 44, o tempo de codificação do grupo 1 é maior que o tempo desejado. Logo, o *PC20* é selecionado para promover um decréscimo no tempo de codificação. O grupo 2 é, então, codificado com o *PC20*. Porém, o tempo de codificação observado no grupo 2 continua superior ao tempo desejado, levando a escolha do *PC24*, ou seja, um *PC* que gera uma RTC ainda maior que o *PC* utilizado na codificação do grupo anterior (*PC20*). Dessa forma, para man-

ter o tempo de codificação de cada grupo próximo ao tempo desejado, os Pontos de Controle são alterados no decorrer da codificação.

Já na Figura 46 apresentada a RTC relacionada a cada grupo de quadros. Nela, os valores dos rótulos representam "grupo; RTC". Como mencionado na Seção 6.3.1, o primeiro grupo de quadros é codificado com o *PC0*, assim sendo, deveria apresentar RTC igual a zero. No entanto, é observado um pequeno valor de RTC, 0,59%, possivelmente acarretado por variações no processo de codificação, como por exemplo, temperatura da memória principal. A partir do grupo 2, a faixa de RTC apresentada varia de 39,56% a 75,35%. Em média a RTC atingida é de 58,25%.

Na Tabela 22 é possível observar os resultados obtidos para a mesma sequência e CQ mencionados anteriormente, porém utilizando o controlador *CCAM*. Assim como na Versão *Baseline* do controlador, os 16 primeiros quadros são codificados sem a atuação controlador, logo na codificação do segundo grupo de quadros, o tempo desejado ajustado (T_{DA}), obtido a partir do tempo desejado e do *budget* de tempo, coincide com o tempo desejado (T_D), pois neste momento o valor do *budget* é zero. O tempo de codificação observado ($T_{GO_{(i)}}$) do primeiro grupo de quadros é obtido e usado para calcular o fator de correção, conforme a Equação 13. Em seguida, com base no tempo estimado pelo modelo de aprendizado de máquina e o fator de correção (Equação 12), o tempo predito ajustado ($T_{GPA_{(i+1)}}$) do grupo é calculado, 173,01 minutos. Posteriormente, a relação entre o tempo desejado ajustado (T_{DA}) e o tempo de predito ajustado ($T_{GPA_{(i+1)}}$) é usado para selecionar o *PC20*. Neste momento, o modelo de classificação já informou ao controlador que a tabela de transição referente a classe *Netflix_SquareAndTimelapse* deve ser usada para buscar o *PC* selecionado. O grupo 2 é, então, codificado com o *PC20*.

Para os grupos seguintes o processo se repete. Porém, o valor do T_{DA} passa a ser, de fato, influenciado pelo *budget*, o qual corresponde ao erro acumulado de todos os grupos codificados anteriormente. Neste exemplo, a RTC média obtida foi de 51,25%, ou seja, um erro de apenas 1,25 pontos percentuais, resultado que é 6,6 vezes mais preciso que a Versão *Baseline* do controlador. Cabe destacar ainda que, nesta sequência, todos os grupos foram codificados com a mesma classe, denominada *Netflix_SquareAndTimelapse*. Porém, no caso da sequência *park_joy_1080p50_60f*, por exemplo, os grupos foram codificados com diferentes classes. A tabela de transição referentes à cada classe pode ser observada no Apêndice F.

Tabela 22 – Resultados obtidos com o controlador *CCAM* para o vídeo Netflix_Boat_1920x1080_60fps_8bit_420_60f, CQ 32, considerando RTC alvo de 50%.

Grupo	T_{CP} (min)	T_{D} (min)	T_{GO} (min)	T_{GPA} (min)	budget (min)	T_{DA} (min)	R_{TA}	PC	Classe	RTC (%)	Média RTC (%)
1	171,83	105,68	170,81	173,01	0,00	105,68	0,61	20	Square	0,6	
2	224,66	105,68	118,33	118,92	-12,64	93,04	0,78	26	Square	47,33	
3	227,08	105,68	57,86	58,13	35,18	140,86	2,42	23	Square	74,52	
4	196,54	105,68	120,20	118,64	20,66	126,35	1,06	23	Square	38,84	
5	199,03	105,68	120,43	121,12	5,92	111,60	0,92	26	Square	39,49	
6	204,24	105,68	52,05	52,10	59,55	165,23	3,17	20	Square	74,51	51,25
7	216,43	105,68	117,81	118,24	47,42	153,11	1,29	15	Square	45,57	
8	192,70	105,68	141,75	140,40	11,36	117,04	0,83	20	Square	26,44	
9	213,71	105,68	118,74	119,58	-1,70	103,98	0,87	23	Square	44,44	
10	217,22	105,68	131,03	131,12	-27,05	78,64	0,6	26	Square	39,68	
11	224,32	105,68	53,32	53,32	25,31	131,00	2,46	23	Square	76,23	

As Figuras 47, 48 e 49 mostram, graficamente, os resultados obtidos com o controlador *CCAM*. A Figura 47, apresenta, além das informações apresentadas na Figura 44 para a Versão *Baseline*, o *budget* (linha rosa), o tempo desejado ajustado (T_{DA}) e o tempo de grupo ajustado ($T_{GPA_{(i+1)}}$), representados pelas linhas pontilhadas roxa e marrom, respectivamente. É possível notar que o tempo desejado (T_{DA}) difere, significativamente, do tempo desejado (T_D), indicando que o *budget* atua expressivamente nas escolhas do controlador. Já os valores de tempo observado de cada grupo ($T_{GO_{(i)}}$) e tempo de grupo predito ajustado ($T_{GPA_{(i+1)}}$) apresentam uma pequena diferença.

Para melhor mensurar o impacto na precisão promovido pelo modelo de predição de tempo de codificação, foi realizada uma análise preliminar considerando três sequências com resolução HD 1080 e uma sequência com resolução UHD 4K, todos com 180 quadros. As sequências HD 1080 usadas foram: *aspen_1080p_60f*, *life_1080p30_60f* e *seaplane_hdr_amazon_1080p*, já a sequência utilizada com resolução UHD 4K foi a *cosmos_aom_9561-9789*. Para cada sequência, foram observados a RTC de 10%, 20%, 30%, 40%, 50%, 60% e 70%, com CQs 20, 32, 43 e 55. Os experimentos foram realizados com a Versão *Baseline* do controlador e com o preditor de tempo de codificação implementado na Versão *Baseline*. Nesta análise, foi possível constatar que o preditor de tempo gera uma melhora na precisão em 3,49% quando comparado a Versão *Baseline* do controlador.

A Figura 48 apresenta os Pontos de Controle selecionados pelo controlador *CCAM* durante a codificação. Ao total, são utilizados quatro *PCs* distintos: *PC15*, *PC20*, *PC23*, *PC26*. Tomando como exemplo o grupo de quadros 6, é possível compreender a dinâmica da seleção dos *PCs* no controlador *CCAM*. O grupo de quadros 6 é codificado com o *PC26*. Na classe "*Netflix_SquareAndTimelapse*", na qual todos os grupos da sequência *Netflix_Boat* são classificados a partir do modelo de aprendizado de máquina, o *PC26* é o que mais impacta na RTC. O uso deste *PC* faz com que o $T_{GPA_{(i+1)}}$ reduza além do necessário. Conforme apresentado na Figura 47, o $T_{GPA_{(i+1)}}$ é 133 minutos menor que o T_{DA}. Para compensar esta diferença, o controlador *CCAM*, então, seleciona para o próximo grupo de quadros o *PC20*, o qual gera menos impacto na RTC se comparado ao *PC26*. Na Figura 49, a qual apresenta a RTC para cada grupo de quadros com o controlador *CCAM*, é possível observar que a RTC do grupo 6 é 74,51%, já no grupo 7 a RTC diminui para 45,57%. Também é possível observar que, assim como na Versão *Baseline* do controlador, o primeiro grupo de quadros apresenta um pequeno valor de RTC, 0,6%. Para os demais grupos, a faixa de RTC varia de 26,44% a 76,23%. Já a RTC média é de 51,25%.

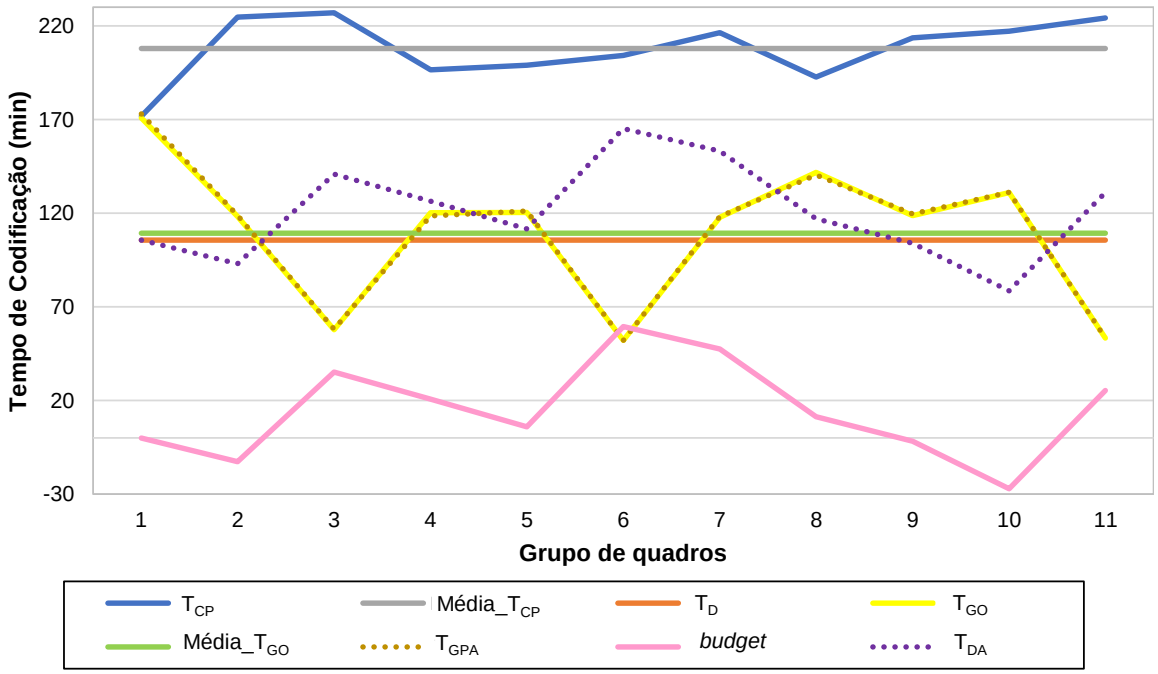


Figura 47 – Resultados obtidos com o controlador *CCAM* para a sequência de vídeo Netflix_Boat_1920x1080_60fps_8bit_420_60f, CQ 32, para a RTC alvo de 50%.

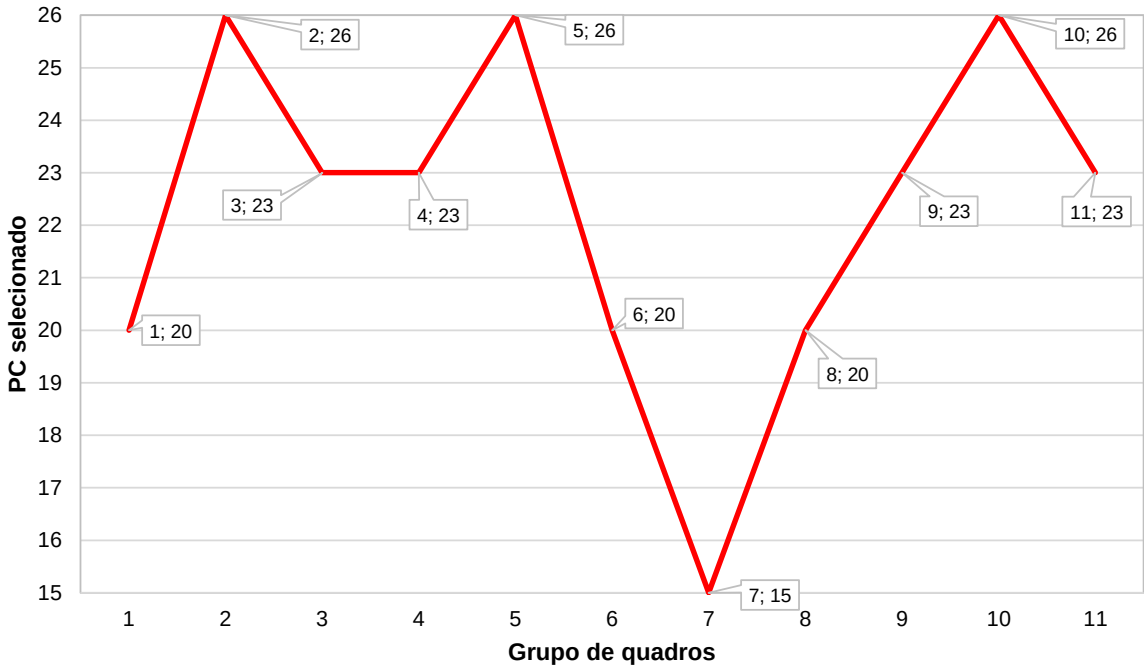


Figura 48 – Próximo Ponto de Controle selecionado, obtido com o controlador *CCAM*, na sequência de vídeo Netflix_Boat_1920x1080_60fps_8bit_420_60f, CQ 32, para a RTC alvo de 50%.

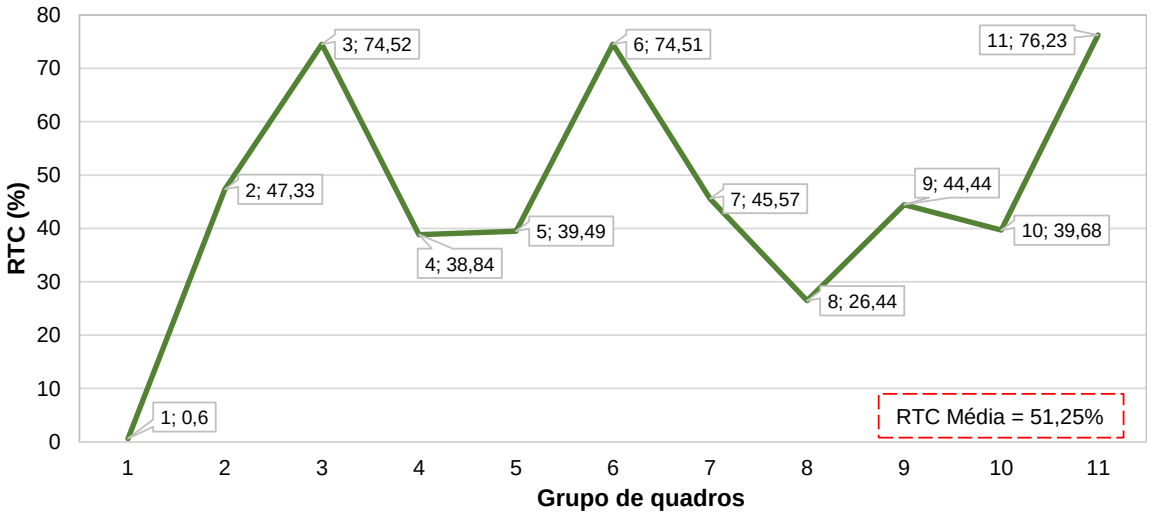


Figura 49 – Redução de tempo de codificação, obtida com o controlador *CCAM*, para a sequência de vídeo Netflix_Boat_1920x1080_60fps_8bit_420_60f, CQ 32, para a RTC alvo de 50%.

7.2.2 Resultados obtidos por CQ e médios: vídeos com resolução HD 1080

Os resultados médios de precisão de cada CQ para a resolução HD 1080 com a Versão *Baseline* do controlador e o controlador *CCAM* são apresentados, respectivamente, nas Tabela 23 e 24. Para as diferentes faixas de RTC analisadas, é possível observar o valor de erro para cada CQ, e a diferença entre o maior e o menor valor de erro observado (coluna "Δ Erro"), possibilitando identificar, para cada faixa de RTC, a máxima variação de precisão existente entre os CQs. Além disso, é mostrado o erro médio de cada CQ em porcentagem. Na grande maioria das faixas de redução de tempo, os valores de erro dos CQs são similares. No Versão *Baseline* do controlador, o maior valor de "Δ Erro" é observado na RTC de 40%, com 16,22%. Já no controlador *CCAM*, a máxima diferença entre os valores de erro dos CQs é de 12,53%, obtido na RTC de 10%. Na Versão *Baseline*, os valores de erro médio apresentados por cada um dos CQs variam de 14,15% a 17%. Já no no controlador *CCAM* esta variação é 6,53% a 8,87%. Cabe destacar que a hipótese de especializar o controlador com base no valor de CQ foi investigada durante esta tese. Porém, os resultados preliminares obtidos, os quais são coerentes aos apresentados nesta seção, indicaram que este tipo de especialização não é vantajosa, pois a diferença de precisão entre os CQs é pequena. Logo, especializar o controlador a partir do valor de CQ não tem grande impacto na melhora da precisão do mesmo.

Tabela 23 – Resultados médios para cada CQ analisado, obtidos com a Versão *Baseline* do controlador de complexidade para as sequências HD 1080 com 180 quadros.

RTC alvo (%)	**Erro (%)**				
	CQ 20	**CQ 32**	**CQ 43**	**CQ 55**	**△ Erro**
10	18,31	30,74	30,47	29,94	12,43
20	17,64	13,27	15,77	13,45	4,37
30	18,75	14,80	13,15	11,76	6,99
40	27,76	15,90	12,59	11,54	16,22
50	14,55	15,15	11,82	9,78	2,72
60	11,86	11,92	7,43	10,15	4,49
70	10,13	11,27	10,36	12,46	2,33
Média	**17,00**	**16,15**	**14,51**	**14,15**	**-**

Tabela 24 – Resultados médios para cada CQ analisado, obtidos com o controlador de complexidade *CCAM* para as sequências HD 1080 com 180 quadros.

RTC alvo (%)	**Erro (%)**				
	CQ 20	**CQ 32**	**CQ 43**	**CQ 55**	**△ Erro**
10	17,46	19,26	28,90	30,00	12,53
20	7,34	8,80	9,60	10,74	3,40
30	2,89	3,69	6,22	4,04	3,33
40	5,21	7,69	5,16	4,73	2,96
50	4,46	3,45	5,13	3,43	1,70
60	3,38	1,25	3,04	2,48	2,13
70	6,11	1,56	4,01	0,85	5,26
Média	**6,69**	**6,53**	**8,87**	**8,04**	**-**

A Tabela 25 apresenta os resultados médios, considerando as cinco sequências HD 1080 com 180 quadros descritas na Seção 7.1 e os quatro CQs avaliados, de precisão e eficiência de compressão para a Versão *Baseline* do controlador e para o controlador *CCAM*. A eficiência de compressão é observada a partir do BD-Rate, já a precisão dos controladores é avaliada com base nos dados presentes na coluna “Erro”, os quais foram obtidos a partir da diferença percentual absoluta entre a redução de tempo de codificação alvo e a redução de tempo de codificação obtida (real). Além disso, na coluna "*overhead*" é possível observar a porcentagem do impacto de cada algoritmo de controle no tempo total de codificação. Cabe destacar que a sequência *speed_bag* não compõe o conjunto de sequências de vídeo HD 1080 definidas pelo IETFNETVC-Testing, por isso, não está sendo considerada nos resultados médios apresentados nesta seção. Os resultados referentes a sequência *speed_bag* e as demais sequências de teste HD 1080 avaliadas são mostrados no Apêndice L.

Tabela 25 – Resultados médios para todas as sequências de teste de resolução HD 1080 com 180 quadros e os quatro CQs avaliados.

RTC alvo (%)	**Versão Baseline**				**CCAM**			
	RTC real (%)	**Erro (± p.p.)**	**BD-Rate (%)**	**Overhead 10^{-6} (%)**	**RTC real (%)**	**Erro (± p.p.)**	**BD-Rate (%)**	**Overhead 10^{-6} (%)**
10	12,74	2,74	0,45	87,50	11,80	1,88	2,08	102,07
20	23,01	3,01	2,53	89,79	20,80	0,80	4,54	108,88
30	34,39	4,39	6,25	107,80	30,11	0,11	8,82	108,04
40	46,78	6,78	8,71	142,42	39,51	0,49	12,95	135,41
50	55,24	5,24	12,38	157,30	49,79	0,21	22,37	151,77
60	65,17	5,17	18,33	193,22	59,41	0,59	30,08	190,08
70	76,16	6,16	41,15	262,16	68,20	1,80	38,30	221,03
Média	**-**	**14,55 (%)**	**12,83**	**148,60**	**-**	**4,05 (%)**	**17,02**	**145,33**

É possível observar, a partir dos resultados apresentados na Tabela 25, que o controlador *CCAM* é significativamente mais preciso que a Versão *Baseline*, com valores de erro variando entre 0,11 e 1,88 pontos percentuais. Para a Versão *Baseline* do controlador, esta faixa varia de 2,74 a 6,78 pontos percentuais. Cabe ressaltar ainda que o *CCAM* obteve tempo de codificação inferior ao RTC alvo nas faixas de 40% a 70%, já o tempo de codificação com a Versão *Baseline* do controlador foi sempre superior ao RTC alvo. O controlador *CCAM* também apresenta BD-Rate máximo, obtido no RTC alvo de 70%, minimamente menor que o obtido com a Versão *Baseline*, 38,30% contra 41,15%. Porém, a Versão *Baseline* do controlador apresenta valores de BD-Rate mais satisfatórios nas demais faixas RTC analisadas. A RTC promovida pela Versão *Baseline* é sempre superior à RTC alvo, já a RTC gerada pelo controlador *CCAM* supera a RTC alvo apenas nas faixas de 10%, 20%, e 30%. Nas demais faixas de RTC, o controlador *CCAM* apresenta a RTC inferior à RTC alvo. Diante disso, é esperado que os valores de BD-Rate da Versão *Baseline* sejam mais elevados que os valores de BD-Rate apresentados pelo controlador *CCAM*, pois a Versão *Baseline* usa Pontos de Controle com maior impacto na RTC e, consequentemente, no valor de BD-Rate. Porém, a Versão *Baseline* apresenta valor de BD-Rate menor que o controlador *CCAM* em seis das sete faixas de RTC analisadas. As prováveis causas deste comportamento, são discutidas a seguir.

Analisando os resultados individuais de cada sequência de vídeo (Apêndice L) é possível observar que nas sequências *Netflix_Boat_1920x1080_60fps_8bit_420_60f*, *park_joy_1080p50_60f* e *rush_hour_1080p25_60f* o controlador *CCAM* apresentou valores de BD-Rate melhores que a Versão *Baseline* do controlador em quatro das sete faixas de RTC analisadas, o que corresponde a, aproximadamente, 57% dos casos. Porém, os resultados médios de BD-Rate são impactados, negativamente, pelas sequências *Netflix_PierSeaside_1920x1080_60fps_8bit_420_60*e *Netflix_FoodMarket_1920x1080_60fps_8bit_420_60f*. Na primeira, o controlador *CCAM* obteve BD-Rate inferior ao obtido pela Versão *Baseline* do controlador apenas na faixa de RTC de 70%, já na segunda sequência não obteve melhor BD-Rate em nenhuma das faixas de RTC analisadas.

Essas duas sequências são as únicas que são classificadas na classe pelo modelo de aprendizado de máquina desenvolvido. Os valores de BD-Rate dos *PCs* 24 e 26 desta classe (Apêndice D), são expressivos, 52,75% e 169,80%, respectivamente. Na codificação da sequência *Netflix_PierSeaside_1920x1080_60fps_8bit_420_60* esses *PCs* foram usados em, aproximadamente, 64% dos experimentos. O *PC26*, especificamente, foi usado em 53% deles. Já na sequência *Netflix_FoodMarket_1920x1080_60fps_8bit_420_60f* estes mesmos *PCs* foram utilizados em 100% dos experimentos realizados, sendo o *PC26* usado em 75% deles. Provavelmente, a grande ocorrência do uso destes *PCs*, prejudicou os resultados de BD-Rate

dessas sequências e, consequentemente, os resultados médios.

Outro fato que, possivelmente, contribuiu para que o controlador *CCAM* obtivesse maiores valores de BD-Rate é a grande variação entre os *PCs* utilizados ao longo da codificação. Variação esta, maior que a observada na Versão *Baseline* do controlador, como visto nas Figuras 45 e 48. Este comportamento pode ser ainda mais acentuado em alguns casos, como mostra a Figura 50, a qual apresenta os *PCs* utilizados na Versão *Baseline* do controlador e no controlador *CCAM* para a sequência *Netflix_FoodMarket_1920x1080_60fps_8bit_420_60f*, CQ 43 e RTC alvo de 40%. Note que a Versão *Baseline* do controlador, além do *PC0*, usa apenas dois Pontos de Controle distintos (*PC12* e *PC15*), os quais apresentam BD-Rate de 0,20% a 0,43% (Tabela 14). Por outro lado, o controlador *CCAM* usa nove Pontos de Controle diferentes, obtidos a partir das classes "*Guitar_Hdr_Amazon*" e "*Netflix_SquareAndTimelapse*" (*PC7*, *PC11*, *PC12*, *PC15*, *PC16*, *PC20*, *PC23*, *PC24* e *PC26*), com BD-Rate de 0,01% a 169,80% (Tabelas 32 e 33 do Apêndice D). Este comportamento, recorrente em grande parte dos experimentos, é influenciado pela significativa diferença de RTC entre os *PCs*. Na classe "*Netflix_SquareAndTimelapse*" por exemplo, o *PC24* tem RTC de 54,06% e BD-Rate de 8,42%. Já o próximo Ponto de Controle, *PC26*, apresenta RTC de 76,99% e BD-Rate de 31,07%. Diante disso, para uma RTC 60%, por exemplo, o controlador seleciona o *PC* com maior impacto no BD-Rate (*PC26*), pois ele gera maior RTC.

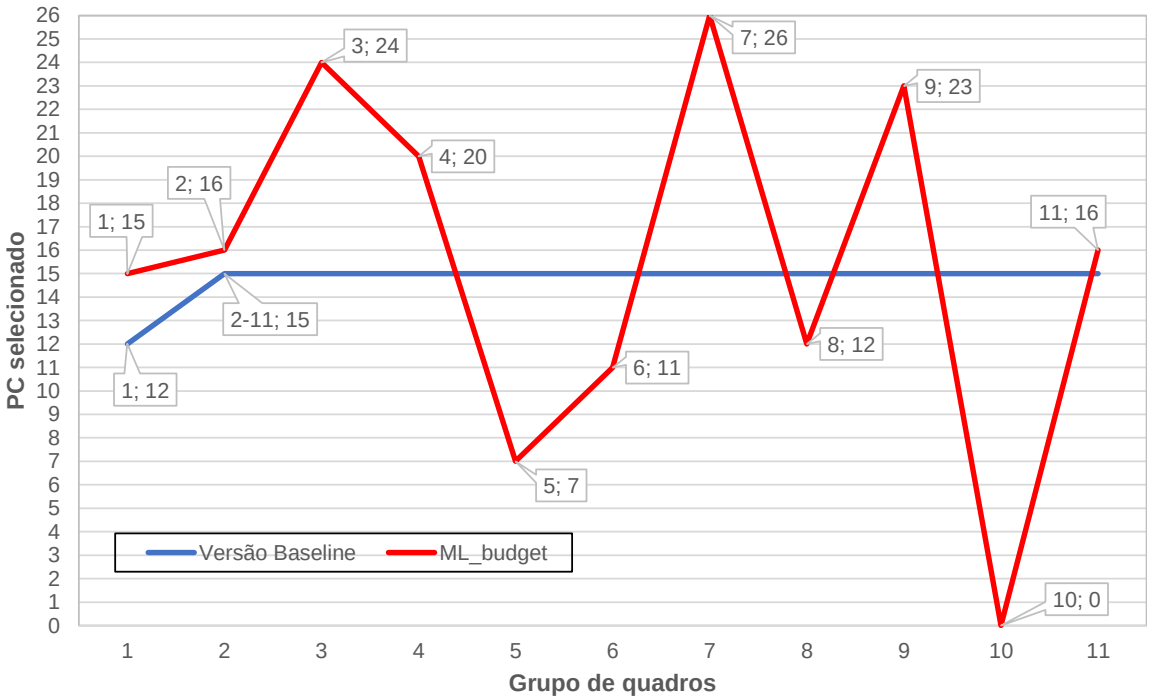


Figura 50 – PCs selecionados para a sequência *Netflix_FoodMarket_1920x1080_60fps_8bit_420_60f*, CQ 43 e RTC alvo de 40%.

Além disso, é possível observar na Tabela 25 que o *overhead* de ambos controladores é extremamente próximo, e pequenos. Para o controlador *baseline*, o *overhead*

médio é 148,60 10^{-6} %. Apesar do controlador *CCAM* utilizar dois modelos de aprendizado de máquina, este valor é ainda, minimamente, menor para o controlador *CCAM*, 145,33 10^{-6} %, indicando que os modelos desenvolvidos e as estratégias de implementação utilizadas não promoveram acréscimo expressivo no tempo total de codificação.

7.2.3 Resultados médios para vídeos com resolução UHD 4K

Os resultados médios, considerando todas as sequências de teste UHD 4K e os quatro CQs avaliados, podem ser observados na Tabela 26. Como esperado, o *overhead* do controlador é menos representativo nas sequências de vídeos com resolução UHD 4K, as quais apresentam tempo de codificação maior quando comparadas aos tempos necessários para a resolução HD 1080. Assim como observado na resolução HD 1080, a precisão do controlador *CCAM* foi, consideravelmente, melhor que a precisão da Versão *Baseline* do controlador, no entanto, os valores de BD-Rate novamente foram superiores na maioria das faixas de RTC.

O erro do controlador *CCAM* varia de 0,14 a 3,33 pontos percentuais, o que demonstra uma variação de 14% em relação aos resultados em HD 1080. Para a Versão *Baseline* do controlador, o erro varia de 3,53 a 6,98 pontos percentuais, correspondendo a uma variação de 44,51% em relação aos resultados em HD 1080. Novamente, o controlador *CCAM* apresenta BD-Rate menor que o obtido com a Versão *Baseline* apenas para a RTC alvo de 70%, 54,57% contra 56,13%. Nas demais faixas analisadas, a Versão *Baseline* do controlador apresenta valores melhores de BD-Rate. Na resolução UHD 4K, a Versão *Baseline* também gera uma RTC sempre superior à RTC alvo, já a RTC obtida pelo controlador *CCAM* supera a RTC alvo apenas na faixa de 70%. Visto que as tendências dos resultados são as mesmas para ambas resoluções, HD 1080 e UHD 4K, é possível afirmar que as discussões realizadas na Seção 7.2.2 também são válidas para os resultados obtidos para a resolução UHD 4K.

Tabela 26 – Resultados médios para todas as sequências de teste de resolução UHD 4K e os quatro CQs avaliados.

RTC alvo (%)	**Versão Baseline**				**CCAM**			
	RTC real (%)	**Erro (± p.p.)**	**BD-Rate (%)**	**Overhead 10^{-6} (%)**	**RTC real (%)**	**Erro (± p.p.)**	**BD-Rate (%)**	**Overhead 10^{-6} (%)**
10	14,95	4,95	0,41	28,13	13,33	3,33	2,86	26,34
20	23,53	3,53	0,90	32,70	22,56	2,56	7,15	27,44
30	34,61	4,61	2,91	37,23	31,06	1,06	10,10	30,44
40	45,99	5,99	5,56	43,51	41,10	1,10	15,33	38,87
50	55,68	5,68	12,07	50,76	50,74	0,74	25,50	45,61
60	66,98	6,98	32,07	64,07	60,82	0,82	43,35	54,47
70	75,75	5,75	56,13	84,65	69,86	0,14	54,57	65,65
Média	**-**	**18,39 (%)**	**15,72**	**48,72**	**-**	**7,91 (%)**	**22,69**	**41,26**

7.2.4 Resultados da variação da RTC em tempo de execução

O experimento com a variação da RTC em tempo de execução necessita de uma quantidade expressiva de quadros para ser realizado. Sequências de teste com apenas 180 quadros não são indicadas para realizar este tipo de experimento, pois, devido a limitação da quantidade de quadros, o controlador não responde de maneira adequada a variação de RTC em tempo de execução. Por isso, para realizar o experimento com a variação da RTC em tempo de execução, foi escolhida a sequência de teste *speed_bag* (XIPH, 2022b), com resolução HD 1080 e um total de 570 quadros, o que corresponde a uma quantidade de quadros três vezes maior que a utilizada nos experimentos anteriores. Os resultados deste experimento foram obtidos a partir do controlador mais preciso, o *CCAM*.

Para a sequência de vídeo *speed_bag* (570 quadros), foram observados resultados referentes ao controlador *CCAM* com a RTC alvo sendo alterada durante a codificação. O primeiro alvo de RTC foi de 10%, foi aplicado aos dezoito primeiros grupos de quadros. Nos 18 grupos restantes foi aplicado a RTC de 30%. Os resultados com o controlador *CCAM* para este experimento, com CQ 55, são mostrados nas Figuras 51, 52 e 53.

A Figura 51 apresenta para cada grupo de quadros os resultados de tempo padrão de codificação T_{CP}, tempo predito ajustado $T_{GPA_{(i+1)}}$, tempo de codificação desejado T_D, *budget*, tempo de codificação observado $T_{GO_{(i)}}$ e tempo desejado ajustado T_{DA}. Além disso, mostra o tempo médio padrão de codificação e o tempo médio do tempo observado. É possível observar que T_{CP} dos grupos 14 ao 19 é muito próximo ao valor do T_{CP} médio, o qual corresponde a 120,87 minutos. Para os demais grupos, o T_{CP} varia em relação ao seu tempo médio em até 50 minutos, aproximadamente. Já o valor do $T_{GO_{(i)}}$ médio corresponde a, aproximadamente, 100 minutos. Há grupos em que o $T_{GO_{(i)}}$ e o $T_{GPA_{(i+1)}}$ são bastante similares. Nos grupos 7 e 8, por exemplo, a diferença entre $T_{GO_{(i)}}$ e o $T_{GPA_{(i+1)}}$ é inferior a um minuto. Porém, em outros grupos de quadros pode ser observada uma diferença bem mais significativa, como nos grupos 9 e 20, por exemplo. Nestes grupos a diferença entre $T_{GO_{(i)}}$ e o $T_{GPA_{(i+1)}}$ é, aproximadamente, 14 minutos.

A Figura 51 também mostra que o tempo desejado T_D para os primeiros 18 grupos de quadro é de, aproximadamente, 109 minutos. Este valor é coerente a RTC desejada para esse conjunto de grupos de quadros, 10%. Nos grupos de quadros 19 a 35, o T_D é alterado para, aproximadamente, 85 minutos, indicando uma RTC alvo de 30%. Além disso, a Figura 51 mostra que, durante todo o período de codificação as curvas relacionadas ao T_{DA} e ao *budget* apresentam comportamento bastante similares, indicando que o T_{DA} é diretamente influenciado pelo *budget* de tempo.

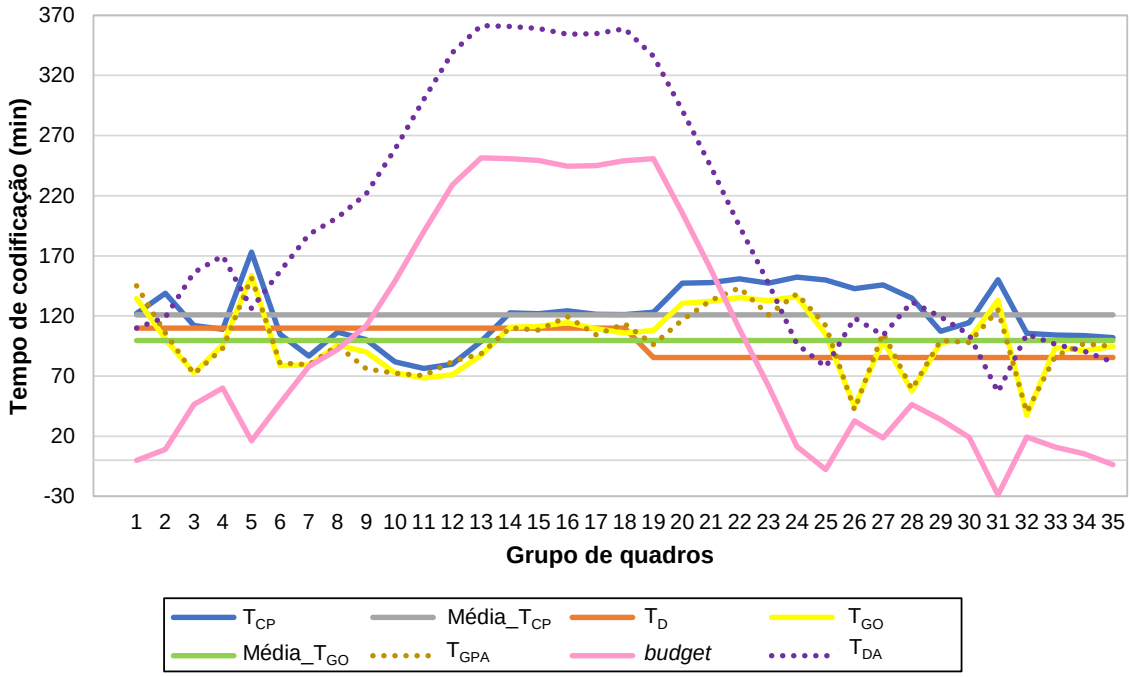


Figura 51 – Resultados obtidos com o controlador *CCAM* para a sequência de vídeo *speed_bag*, CQ 55, para a RTC alvo de 10%-30%.

A Figura 52 apresenta os Pontos de Controle selecionados por cada grupo de quadros. Para atender a RTC alvo, a partir das classes "*Guitar_Hdr_Amazon*", "*Netflix_SquareAndTimelapse*"e "*Netflix_TunnelFlag*", o controlador *CCAM* escolhe oito Pontos de Controle diferentes, são eles: *PC0*, *PC8*, *PC12*, *PC15*, *PC19*, *PC21*, *PC24* e *PC26*. É possível observar que o *PC0* é selecionado consecutivamente pelos grupos de 6 a 23. Isso ocorre porque nesses grupos a redução do tempo de codificação não é necessária, pois o valor do $T_{GPA_{(i)}}$ é sempre menor que o valor do T_{DA} (Figura 51). Dessa forma, o controlador *CCAM* seleciona o único Ponto de Controle que não promove redução do tempo de codificação, o *PC0*.

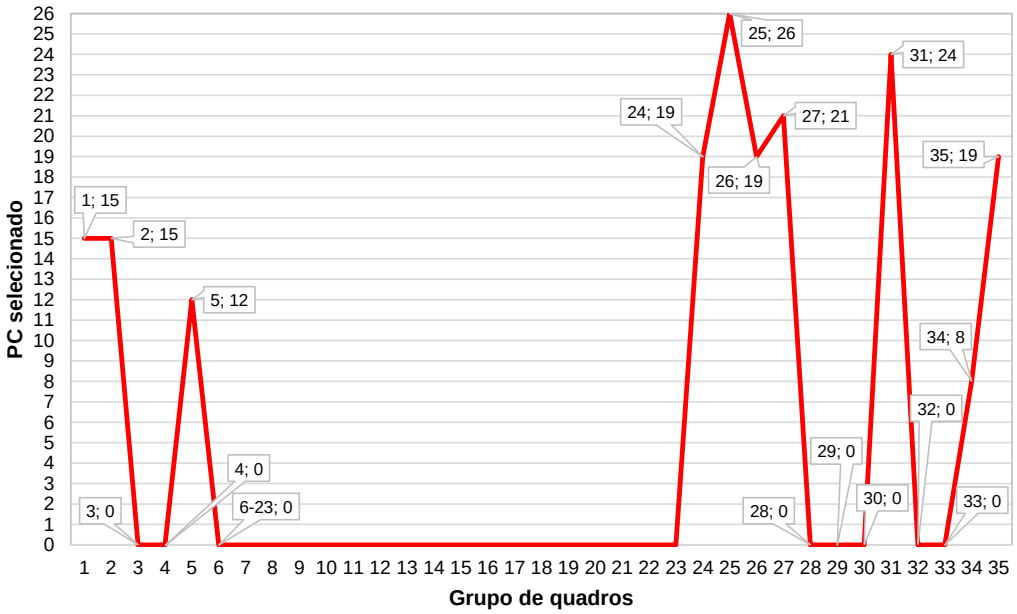


Figura 52 – Próximo Ponto de Controle selecionado, obtido com o controlador *CCAM*, na sequência de vídeo *speed_bag*, CQ 55, para a RTC alvo de 10%-30%.

Na Figura 53 é apresentado o impacto na RTC de cada grupo de quadros analisado. É possível observar que o controlador *CCAM* responde adequadamente ao *PC* selecionado. O grupo de quadros 7 ao 24, por exemplo, os quais são codificados *PC0*, apresentam pequenos valores de RTC. Por outro lado, os grupos de quadro 26 e 32 apresentam os maiores valores de RTC, 68,64% e 64,66%, respectivamente. O grupo 26 é codificado com o *PC26*, já o grupo 32 utiliza *PC24*, ou seja, os dois *PCs* mais severos em termos de redução de tempo de codificação.

Para a RTC alvo de 10% o erro obtido foi de 2,38 pontos percentuais. Já na RTC alvo de 30% este valor foi de 7,68 pontos percentuais. Estes valores são maiores que o erro apresentado pelo controlador *CCAM* nas faixas de RTC alvo de 10% e 30% para a sequência de teste *speed_bag*, porém, sem alteração da RTC alvo durante a codificação. Para as RTC alvos de 10% e 30% o erro obtido foi de, respectivamente, 1,06 e 0,83 pontos percentuais, como mostrado no Apêndice L. Ao comparar os resultados, é possível observar que os Pontos de Controle utilizados pelo controlador com RTC fixo e alterado durante a codificação não são os mesmos, nem mesmo durante a codificação dos primeiros 18 quadros, como era esperado. Apesar da tentativa de gerar condições de testes iguais para todos os experimentos, mantendo sempre os núcleos ocupados, houve diferença no tempo de codificação dos grupos. Fatores como, por exemplo, tempo de acesso à memória principal e a cache, uso da CPU e temperatura da memória podem ter influenciado no tempo de codificação de cada experimento, levando a seleção de *PCs* diferentes e, consequentemente, a obtenção de valores de erros distintos.

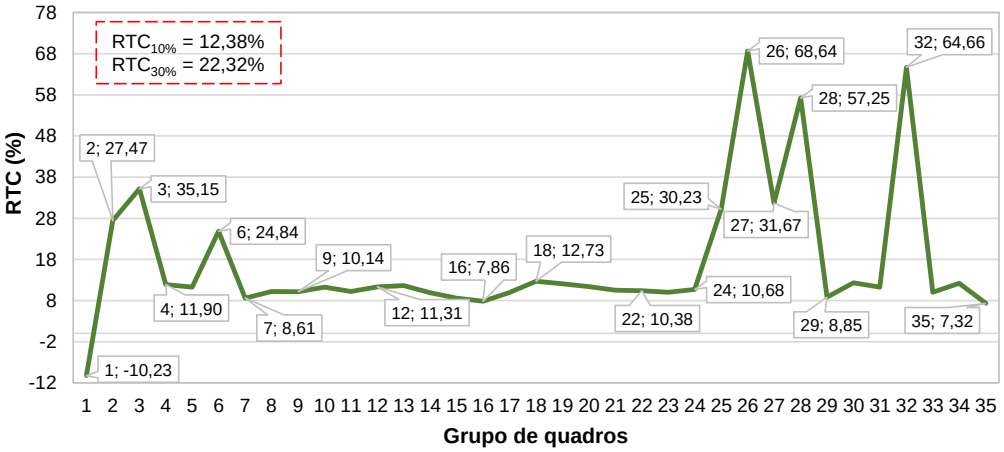


Figura 53 – Redução de tempo de codificação, obtida com o controlador *CCAM*, para a sequência de vídeo *speed_bag*, CQ 55, para a RTC alvo de 10%-30%.

7.2.5 Comparações com trabalhos relacionados

Como já mencionado anteriormente, até o presente momento, não foi encontrado nenhum controlador de complexidade para o codificador AV1 na literatura. Este fato impossibilita que o controlador desenvolvido nesta tese, o *CCAM*, seja comparado com outro controlador específico para o AV1. No entanto, uma comparação pode ser realizada em relação a controladores de complexidades desenvolvidos voltados para outros codificadores de vídeo, como o HEVC e o VVC.

A Tabela 27 apresenta os principais resultados obtidos com o controlador desenvolvido nesta tese, o *CCAM*, além dos resultados referentes a quatro controladores de complexidade encontrados na literatura, sendo três controladores desenvolvidos para o HEVC (DENG et al., 2016), (ZHANG et al., 2018a) e (HUANG et al., 2021a), e um controlador para o VVC (HUANG et al., 2022). Na Tabela 27, para cada controlador, são apresentados os resultados de erro médio e BD-Rate. Além disso, são observadas as faixas de RTC, os CQs/QPs, o codificador e o software de referência utilizados nos experimentos.

O erro médio apresentado pelo controlador *CCAM* varia de 0,11 a 3,33 pontos percentuais. Estes resultados são melhores que os obtidos pelos controladores de complexidade abordados em (HUANG et al., 2022) e (HUANG et al., 2021a). Em Huang et al. (2022), é apresentado um controlador de complexidade para o VVC que trabalha na mesma faixa de RTC do controlador *CCAM* (10% a 70%). O erro obtido foi de 0,04 a 12,22 pontos percentuais. Já em Huang et al. (2021a), o erro está entre 1,7 e 5,6 pontos percentuais, porém a faixa de RTC observada para o codificador HEVC é reduzida, 40% a 60%.

Os valores de BD-Rate apresentados pelo controlador *CCAM* variam de 2,08% a 54,57%. Entre os controladores de complexidade encontrados na literatura para os codificadores HEVC e VVC, o máximo valor de BD-Rate observados é de, aproxima-

damente, 18%. Os controladores complexidade apresentados por (DENG et al., 2016) e (ZHANG et al., 2018a), trabalham na faixa de RTC de 20% a 80%. Os valores de BD-Rate apresentados em (DENG et al., 2016) variam de 0,84% a 17,49%, já os obtidos em (ZHANG et al., 2018a) variam de 1,36% a 18,11%.

Cabe salientar que não é possível realizar uma comparação direta dos resultados do controlador desenvolvido nesta tese e com os resultados dos trabalhos da literatura. Os codificadores diferem em diversos aspectos, tais como: estrutura de particionamento, modos de predição intra-quadro e inter-quadros, tipos de transformadas e entropia (SULLIVAN et al., 2012) (SALDANHA et al., 2020). Também, não existe relação direta entre os CQs/QPs (BENDER et al., 2019). Outro fator que prejudica a comparação, é que os resultados foram obtidos com base em um conjunto diferente de sequências de teste, uma vez que cada codificador tem uma CTC (*Common Test Coditions*), e que as escolhas dos vídeos dentro desses grupos também foram diferentes em cada trabalho. Além disso, os trabalhos usam diferentes faixas de RTC alvo.

Tabela 27 – Comparação entre o controlador *CCAM* e outros controladores encontrados na literatura.

Controlador	**Codec/ SW de Referência**	**Faixa de RTC (%)**	**Faixa de Erro médio (± p.p.)**	**Faixa de BD-Rate (%)**	**BD-Rate/RTC**	**CQs/QPs**
CCAM	AV1/ Libaom 2.0	10 - 70	0,11 - 3,33	2,08 - 54,57	0,21 - 0,78	20,32,43,55
(HUANG et al., 2022)	VVC/ VTM 10.0	10 - 70	0,04 - 12,22	0,23 - 2,71	0,02 - 0,04	22,27,32,37
(DENG et al., 2016)	HEVC/ HM 14.0	20 - 80	0,73 - 2,54	0,84 - 17,49	0,04 - 0,22	22,27,32,37
(ZHANG et al., 2018a)	HEVC/ HM 16.9	20 - 80	0,01 - 1,18	1,36 - 18,11	0,07 - 0,23	22,27,32,37
(HUANG et al., 2021a)	HEVC/ HM 16.5	40 - 60	1,7 - 5,6	0,07 - 1,9	0,00 - 0,03	22,27,32,37

7.3 Testes de Avaliação de Qualidade Subjetiva

O teste de avaliação de qualidade subjetiva é a única maneira de avaliar a percepção visual humana em relação às degradações na qualidade do vídeo. Diante disso, a avaliação de qualidade subjetiva é realizada para verificar se o observador percebe, visualmente, a perda de qualidade gerada por diferentes faixas de reduções de complexidade. No contexto desta tese, esta análise é importante devido aos elevados resultados de BD-Rate observados, em algumas sequências, para faixas mais elevadas de RTC alvo. A partir da avaliação subjetiva é possível identificar se os valores elevados de BD-Rate também causam degradações significativas nas imagens para a visão humana.

Devido a similaridade entre os resultados objetivos de ambas resoluções analisadas nesta tese (HD 1080 e UHD 4K), os testes subjetivos foram realizados considerando apenas sequências de vídeo com resolução HD 1080. Os testes subjetivos foram realizados com base nas recomendações encontradas em (ITU-R, 2012) e (ITU-T, 1999). As cinco sequências de vídeo HD 1080 usadas nos testes objetivos (Subseção 7.1) foram analisadas, com CQ 32, a partir do método de avaliação DCR (*Degradation Category Rating*), ilustrado na Figura 54. Neste método as sequências de teste devem ser apresentadas em pares: o primeiro estímulo apresentado em cada par é sempre a referência (sequência codificada com a configuração padrão) enquanto o segundo estímulo é a mesma sequência, porém com a utilização do controlador *CCAM* em uma RTC alvo. Dessa forma, para cada sequência de vídeo foi analisado o impacto promovido pelo controlador *CCAM* nas RTC de 10%, 30%, 50% e 70%. O impacto na qualidade subjetiva foi medido a partir da métrica MOS (*Mean Opinion Score*), a qual quantifica o impacto na qualidade subjetiva a partir de valores crescentes de 0 a 5. O valor zero indica que a perda de qualidade é muito perceptível, já o valor 5 aponta que a perda de qualidade é imperceptível.

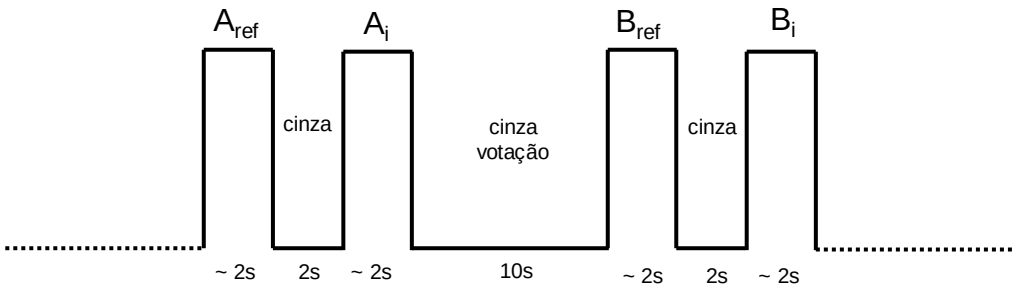


Figura 54 – Apresentação do estímulo no método DCR. Adaptado de (ITU-R, 2012).

O *setup* foi montado em uma sala situada no curso Técnico de Eletrônica do Instituto Federal Sul-rio-grandense (Campus Pelotas). Este fato teve influência no perfil

dos sujeitos. Ao total foram 20 avaliadores, 10 estudantes e 10 professores, entre 18 e 56 anos como mostra a Figura 55. Na Figura 56 é possível observar que 60% dos avaliadores não utilizam óculos e 40% usam. Entretanto, é importante salientar que todas as avaliações foram antecedidas por um teste de acuidade visual e um teste de daltonismo (ITU-R, 2012). No teste de acuidade visual, a tabela Snellen, Anexo A, foi fixada no monitor usado para os testes. Já no teste de daltonismo foi realizado através de um aplicativo de celular, o qual gera os resultados com base em 39 placas Ihihara (INC., 2022). Os observadores que não cometeram erros na linha 20/30 da tabela Snellen e não foram diagnosticados como possíveis daltônicos, foram considerados aptos para realizar os testes.

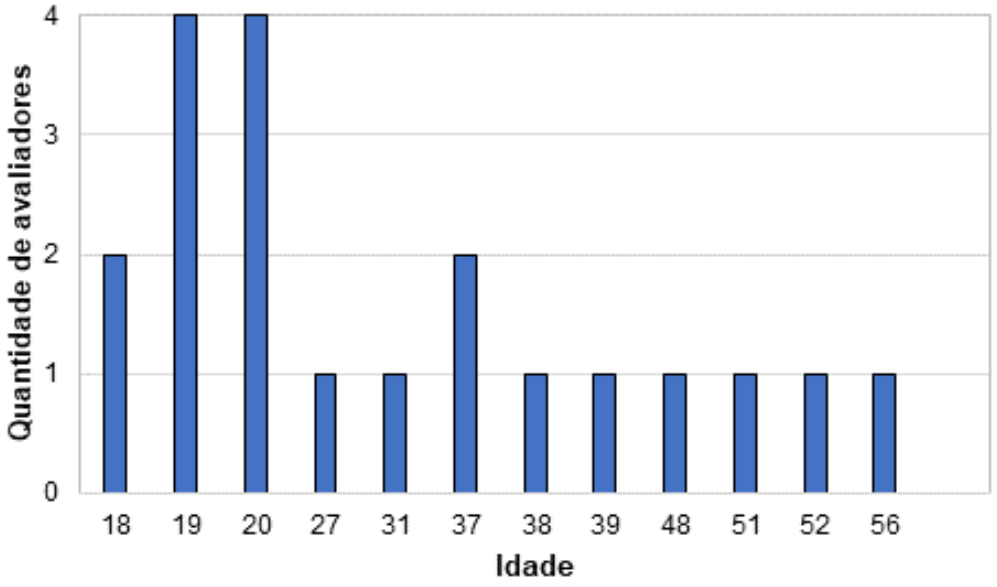


Figura 55 – Faixa etária dos avaliadores.

As sequências de vídeo foram visualizadas através de uma televisão 32 polegadas, *Full HD*. Com base na altura do monitor, os avaliadores foram mantidos a uma distância de 1,20 metros do mesmo, conforme recomendado em (ITU-R, 2012). Além disso, a intensidade da luz do ambiente foi monitorada durante o decorrer do processo, com auxílio de um luxímetro. A cada sujeito foi entregue um documento contendo o procedimento de teste e a ficha de avaliação. Os procedimentos de teste também foram explicados verbalmente e o avaliador passou por uma etapa de treinamento. Posteriormente, os avaliadores analisaram o conjunto de teste proposto e, para cada teste observado, documentaram suas opiniões. As sequências de testes, assim como o documento com o procedimento de teste e a ficha de avaliação podem ser observados no Apêndice K.

Os dados coletados foram analisados a partir de uma técnica avançada de análise de dados, descrita em (ITU-T, 1999). Esta técnica, implementada em Phyton, estima o MOS de cada sequência de vídeo processada, modelando comportamentos sub-

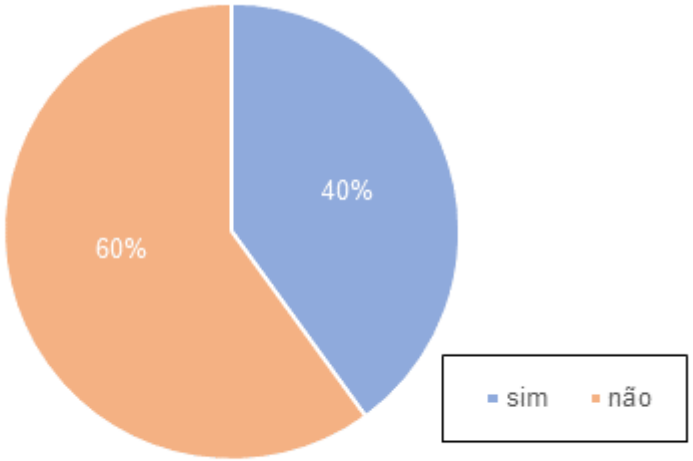


Figura 56 – Perfil dos avaliadores em relação a utilização de óculos de grau.

jetivos como a consistência do avaliador e opiniões influenciadas por algum tipo de preconceito, por exemplo, se o observador gostou ou não da sequência de referência. Para um sujeito que vota inconsistentemente, por exemplo, os votos têm um peso pequeno, contribuindo pouco para o MOS geral.

7.3.1 Resultados da Avaliação de Qualidade Subjetiva

Na Figura 57 é possível ver, para cada faixa de RTC analisada no teste subjetivo, os resultados da avaliação de qualidade subjetiva realizada a partir das cinco sequências de vídeo HD 1080 analisadas. Os resultados mostram que, na opinião dos avaliadores, o comprometimento na qualidade subjetiva das sequências comprimidas em relação a sequências sem compressão é, praticamente, imperceptível. A média da pontuação MOS é 4,64. Como esperado o valor de MOS decresce à medida a RTC aumenta. O menor valor de MOS observado é de 4,49 e corresponde a RTC alvo de 70%. Já o maior valor de MOS, 4,81, é obtido com a RTC alvo de 10%. A Figura 58 apresenta, o SOS (*Standard Deviation of Score*) referente a pontuação observada na Figura 57. O valor médio de SOS é 0,16, já o máximo valor de SOS é 0,21 e é observado para a RTC alvo de 50%.

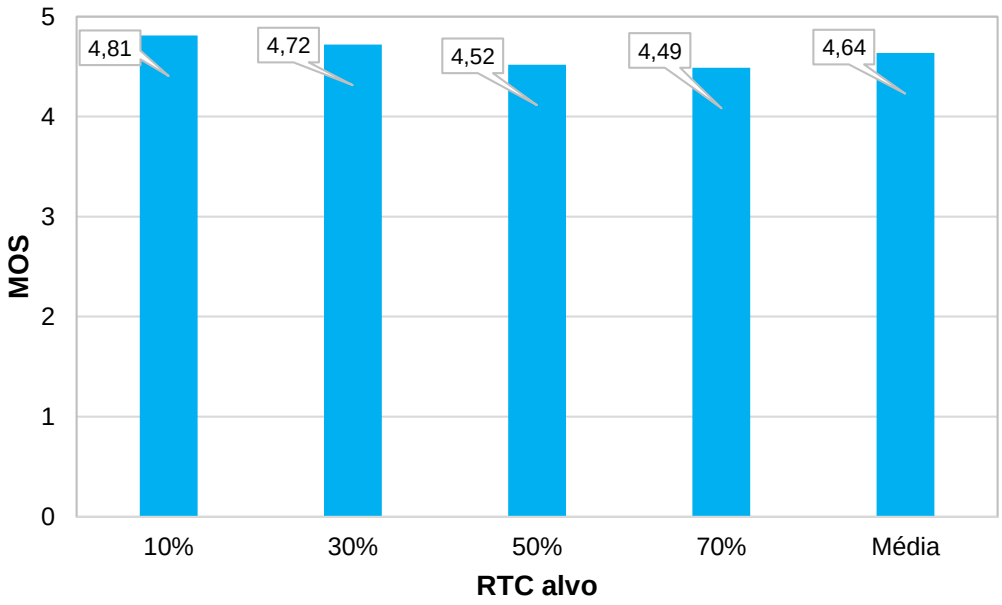


Figura 57 – Resultado MOS das sequências de vídeo com resolução HD 1080.

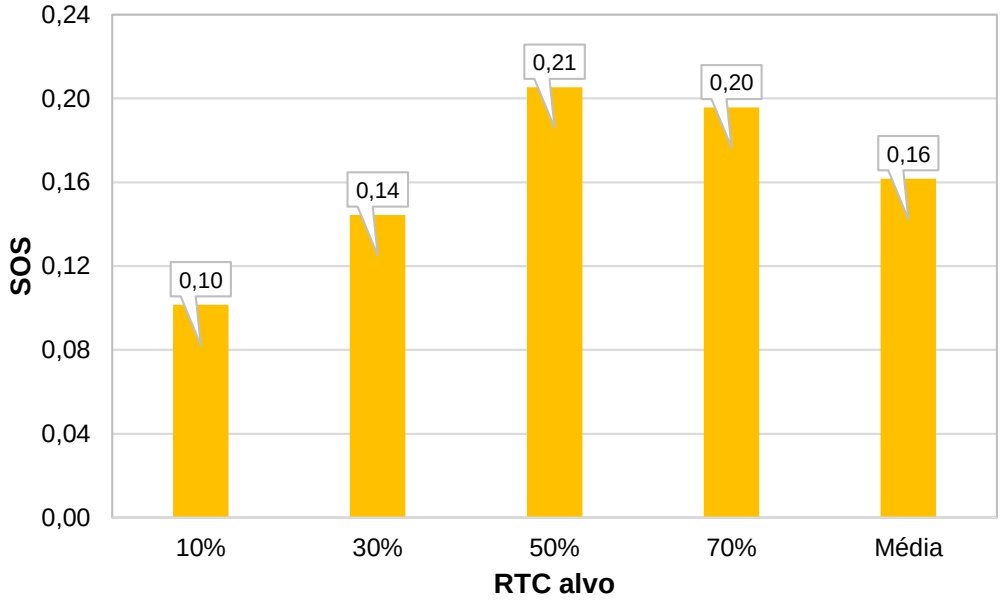


Figura 58 – Resultado SOS das sequências de vídeo com resolução HD 1080.

O valor de MOS de cada sequência HD 1080 analisada é apresentada na Figuras 59. As sequências *Netflix_Boat_1920x1080_60fps_8bit_420_60f*, Netflix_PierSeaside_1920x1080_60fps_8bit_420_60f e *rush_hour_1080p25_60f* compartilham a menor pontuação MOS, 4,56. Por outro lado, a maior pontuação MOS está relacionada a sequência *park_joy_1080p50_60f* e corresponde a 4,75. Na Figura 60 são apresentados os valores de SOS de cada sequência analisada. Nela, é possível ver que as sequências Netflix_FoodMarket_1920x1080_60fps_8bit_420_60f e *park_joy_1080p50_60f* mostram o menor valor de SOS obtido, 0,12. Já a sequência *rush_hour_1080p25_60f* apresenta SOS igual a 0,18, o que corresponde ao maior valor de SOS observado.

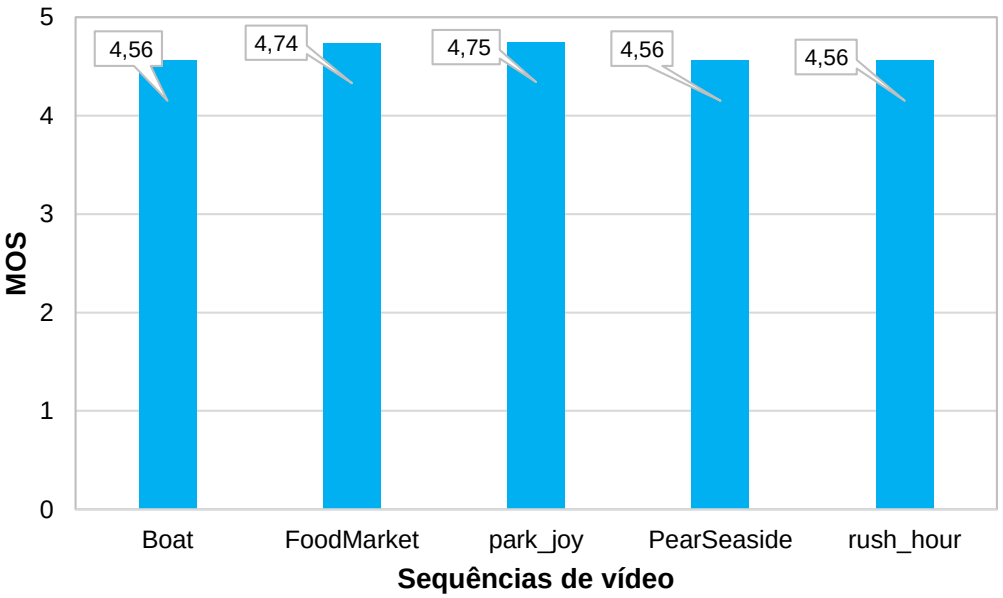


Figura 59 – Resultado MOS das sequências de vídeo com resolução HD 1080.

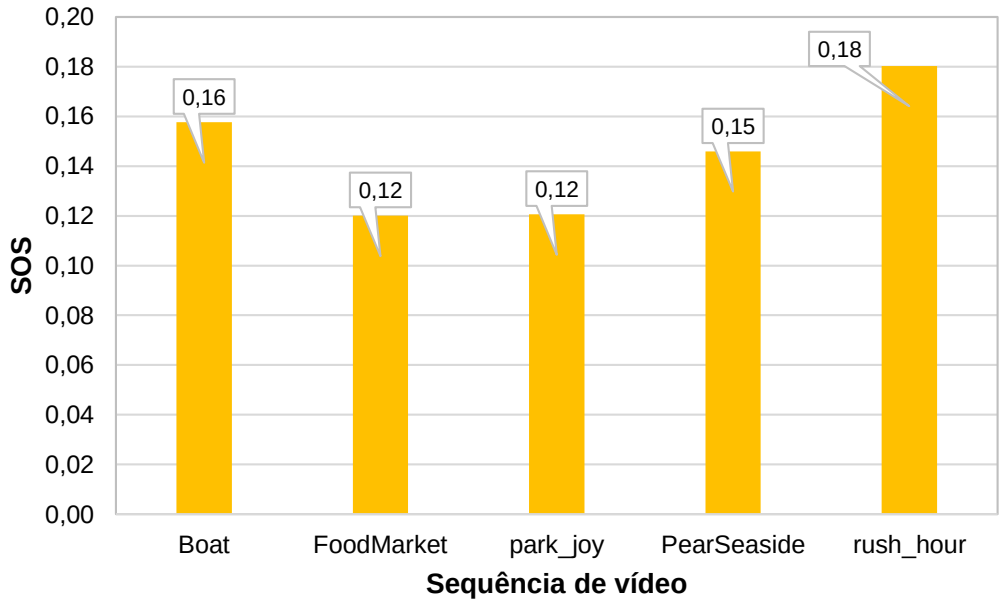


Figura 60 – Resultado SOS das sequências de vídeo com resolução HD 1080.

Para melhor explorar os resultados da avaliação de qualidade subjetiva, a Tabela 28 apresenta os resultados objetivos e subjetivos referentes as cinco sequências HD 1080 analisadas no teste subjetivo, considerando as quatro faixas de RTC alvo e o CQ 32, utilizados nos testes subjetivos. A Tabela 28 apresenta os resultados médios de Δ PSNR e Δ *bitrate*, aos resultados objetivos de eficiência de codificação (BD-Rate) e os valores de MOS obtidos na avaliação subjetiva. A sequência *park_joy_1080p50_60f*, por exemplo, apresenta a menor perda observada tanto na qualidade subjetiva, pontuação MOS de 4,75, quanto na qualidade objetiva, Δ PSNR igual a -0,09 dB. Porém, os resultados da sequência *rush_hour_1080p25_60f* indicam que nem sempre é possível traçar uma relação direta entre qualidade objetiva e qualidade subjetiva. A sequência *rush_hour_1080p25_60f* também apresenta a menor perda de qualidade objetiva observada, entretanto está relacionada com o maior grau de degradação de qualidade apontado pelos observadores, o que corresponde ao valor MOS de 4,56. A partir dos resultados da Tabela 28 também é possível observar que o impacto na qualidade objetiva é pequeno, variando de -0,09 dB a -0,14 dB. Por outro lado, a taxa cresce muito menos que os valores de BD-Rate obtidos, indicando que as sequências usadas nos testes subjetivos não são tão diferentes em tamanho, como o valor de BD-Rate poderia sugerir.

Tabela 28 – Resultados médios para as sequências de vídeo HD 1080 analisadas.

Sequências	**Resultados Objetivos**			**Resultado Subjetivo**
	CQ 32		**BD-Rate médio (%)**	**Pontuação MOS**
	$\triangle$ PSNR médio (dB)	**$\triangle$ bitrate médio (%)**		
Boat	-0,13	3,46	6,87	4,56
FoodMarket	-0,13	14,28	35,24	4,74
park_joy	-0,09	2,67	5,77	4,75
PearSeaside	-0,14	7,87	18,85	4,56
rush_hour	-0,09	7,03	22,74	4,56

8 CONCLUSÕES

Esta tese teve como objetivo o projeto de um controlador adaptativo de complexidade para o codificador de vídeo AV1 com foco em vídeos de resolução HD 1080 e UHD 4K, explorando o uso de Frente de Pareto e aprendizado de máquina. As características do codificador AV1 foram discutidas e uma análise de um conjunto de suas ferramentas foi apresentada, considerando os impactos em tempo e eficiência de codificação. Um estudo da literatura atual demonstrou que existem muitos trabalhos relacionados com a redução de complexidade para o AV1 e com o controle de complexidade para outros padrões de codificação. Entretanto, cabe destacar que não existem controladores de complexidade específicos ao AV1 na literatura.

Para a compreender a complexidade do AV1, diferentes análises foram realizadas. Uma delas, foi a análise de perfil de complexidade que permite entender os esforços computacionais exigidos pelas diferentes etapas do processo de codificação AV1. Outra, foi a análise da estrutura de particionamento para observar como as decisões de particionamento ocorrem no AV1. Também foi feita uma análise de sensibilidade do AV1 em relação aos parâmetros de codificação para observar o impacto no custo computacional e eficiência de codificação promovido por esses parâmetros. Essas análises são apresentadas e discutidas no Capítulo 5.

A metodologia adotada consistiu em desenvolver uma Versão *Baseline* do controlador, adaptando ao codificador AV1 uma solução de controle de complexidade já existente na literatura. Depois, a partir da Versão *Baseline*, foi desenvolvido um controlador otimizado, explorando a especialização e o aprendizado de máquina. Este controlador, denominado de *CCAM*, utiliza dois modelos aprendizado de máquina: um para realizar a predição do tempo de codificação e outro para a especializar o controle de acordo com as características do vídeo de entrada.

Os resultados mostram que, para a resolução HD 1080, a Versão *Baseline* do controlador apresenta um erro que varia de 2,74 a 6,78 pontos percentuais com BD-Rate de 0,45% a 41,15%, para faixas de redução de complexidade variando de 10% a 70%. Considerando todas as faixas de RTC alvo analisadas, o erro médio é de 14,55% e o BD-Rate médio obtidos é 12,83%. Para resolução UHD 4K esses valores

são maiores, sendo que o erro varia de 3,53 a 6,98, com BD-Rate de 0,41% a 56,13% para as mesmas faixas de redução de complexidade. Neste caso, o erro médio é de 18,39% e o BD-Rate médio é de 15,72%. É possível observar que o controlador *CCAM* é, significativamente, mais preciso que a Versão *Baseline*. Para a resolução HD 1080, o erro varia de 0,11 a 1,88 pontos percentuais e o BD-Rate varia de 2,08% a 38,30%. O erro médio obtido a partir do controlador *CCAM* é de 4,05% com BD-Rate médio de 17,02%. Já para a resolução UHD 4K, a faixa de erro é de 0,14 a 3,33 pontos percentuais e a faixa de BD-Rate é de 2,86% a 54,57%. O erro médio apresentado é de 7,91% com BD-Rate médio de 22,69%. Além disso, esta tese também apresentou uma análise de qualidade subjetiva do controlador *CCAM*. Os resultados mostram que a degradação na qualidade visual com a aplicação do controlador é praticamente imperceptível e o impacto na qualidade objetiva é pequeno. Por outro lado, a taxa cresce muito menos que os valores de BD-Rate obtidos, indicando que as sequências usadas nos testes subjetivos não são tão diferentes em tamanho, como o valor de BD-Rate poderia sugerir.

Como já mencionado anteriormente não existe, na literatura atual, outro controlador de complexidade para ao codificador AV1, inviabilizando, portanto, uma comparação justa com os resultados obtidos nesta tese. No entanto, uma comparação foi realizada com controladores desenvolvidos para outros padrões de codificação. Em (HUANG et al., 2022), o erro obtido, a partir de um controlador de complexidade para o VVC com RTC de 10% a 70%, é de 0,04 a 12,22 pontos percentuais. Já em (HUANG et al., 2021a), o erro varia de 1,7 e 5,6 pontos percentuais. A faixa de RTC deste trabalho é de 40% a 60%. É válido destacar que os resultados não são diretamente comparáveis, pois os codificadores são distintos, por exemplo, na estrutura de particionamento, nos modos de predição, nos tipos de transformadas e na entropia (SULLIVAN et al., 2012) (SALDANHA et al., 2020). Além disso, não há relação direta entre os CQs/QPs (BENDER et al., 2019), usam *setups* de experimentos diferentes e não analisam as mesas faixas de RTC alvo.

Como trabalhos futuros, espera-se desenvolver otimizações que melhorarem a eficiência de codificação do controlador. Levando em consideração que grande parte dos trabalhos da literatura reduzem a complexidade explorando a árvore de particionamento dos codificadores em questão, esta abordagem também poderia ser utilizada para gerar Pontos de Controle com RTC intermediários aos já existentes. Um novo estudo também pode ser realizado para identificar parâmetros com menores perdas na eficiência de codificação e na reaplicação da metodologia adotada nesta tese para selecionar os novos Pontos de Controle para o controlador.

REFERÊNCIAS

AKRAMULLAH, S. **Digital Video Concepts, Methods and Metrics**: Quality, Compression, Performance and Power Trade-off Analysis. [S.l.]: Springer Nature, 2014.

AOMEDIA. 2016. Disponível em: <https://aomedia.googlesource.com/aom/+/refs/heads/main/av1/encoder/speed_features.h>. Acesso em: outubro de 2021.

AOMEDIA. **AV1 Codec Library**. 2018. Disponível em: <https://aomedia.googlesource.com/aom/>. Acesso em: outubro de 2020.

AOMEDIA. **AV1 features**. 2018. Disponível em: <https://aomedia.org/av1-features/>. Acesso em: outubro de 2018.

AOMEDIA. **He Open and Royalty-Free Codec for Next-Generation Ultra High Definition Media**. 2019. Disponível em: <https://aomedia.org>. Acesso em: fevereiro de 2019.

AOMEDIA. **They Developed It. They Benefit From It. They Stand Behind It**. 2022. Disponível em: <https://aomedia.org/membership/members/>. Acesso em: maio de 2022.

ARMASU, L. **Releases Open Source Encoder for Nest-Gen AV1 Codec**. 2019. Disponível em: <https://www.tomshardware.com/news/intel-svt-av1-open-source-encoder,38551.html>. Acesso em: julho de 2019.

AYADI, L. A. et al. HEVC Decoder Analysis on ARM Processor. In: IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA AND EXPO (ICME), 2018. **Anais...** [S.l.: s.n.], 2018. p.842–845.

BENDER, I. et al. Compression Efficiency and Computational Cost Comparison between AV1 and HEVC Encoders. In: IEEE EUROPEAN SIGNAL PROCESSING CONFERENCE (EUSIPCO), 2019. **Anais...** [S.l.: s.n.], 2019. p.1–5.

BITMOVIN. **On demand kickoff webinar**: Covid-19 and the impact on Ott Video. 2020. Disponível em: <https://go.bitmovin.com/covid19-ott-video-impact>. Acesso em: abril de 2022.

BJØNTEGAARD, G. Calculation of average PSNR differences between RD-curves. **VCEG-M33**, [S.l.], 2001.

BOSSEN, F. et al. Common test conditions and software reference configurations. **JCTVC-L1100**, [S.l.], v.12, n.7, 2013.

BROSS, B.; CHEN, J.; LIU, S.; WANG, Y.-K. **Versatile Video Coding Editorial Refinements on Draft 10**. 2020. Disponível em: <https://jvet-experts.org/doc_end_user/current_document. php?id=10540>. Acesso em: abril de 2022.

BURKOV, A. **The hundred-page machine learning book**. [S.l.]: Andriy Burkov Quebec City, QC, Canada, 2019. v.1.

BöTTGER, T.; IBRAHIM, G.; VALLIS, B. How the internet reacted to covid-19: A perspective from facebook's edge network. In: IEEE INTERNET MEASUREMENT CONFERENCE (IMC), 2020. **Anais...** [S.l.: s.n.], 2020. p.34–41.

CARVALHO NOBRE PALAU, R. de. **Investigação de Soluções em Hardware para os Filtros de Laço do Decodificador AV1**. 2022. 104p. Tese — universidade Federal de Pelotas, Pelotas.

CHEN, F. et al. Hierarchical complexity control algorithm for HEVC based on coding unit depth decision. **EURASIP Journal on Image and Video Processing**, [S.l.], n.96, p.1–14, 2018a.

CHEN, G. et al. AV1 in-loop Filtering using a Wide-Activation Structured Residual Network. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2019a. **Anais...** [S.l.: s.n.], 2019a. p.1725–1729.

CHEN, X. et al. A Conditional Bayesian Block Structure Inference Model for Optimized AV1 Encoding. In: IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA AND EXPO (ICME), 2019. **Anais...** [S.l.: s.n.], 2019. p.1270–1275.

CHEN, X.; ZHANG, Y.; LI, Y.; WEN, J. Decision Tree Based Inter Partition Termination for AV1 Encoding. In: IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING (ICASSP), 2021. **Anais...** [S.l.: s.n.], 2021. p.1585–1589.

CHEN, Y. et al. An Overview of Core Coding Tools in the AV1 Video Codec. In: IEEE PICTURE CODING SYMPOSIUM (PCS), 2018. **Anais...** [S.l.: s.n.], 2018. p.41–45.

CHEN, Y. et al. An Overview of Coding Tools in AV1: The First Video Codec from the Alliance for Open Media. **APSIPA Transactions on Signal and Information Processing**, [S.l.], v.9, p.1–15, 2020.

CHIANG, C. H.; HAN, J.; XU, Y. A Multi-Pass Coding Mode Search Framework for AV1 Encoder Optimization. In: IEEE DATA COMPRESSION CONFERENCE (DCC), 2019. **Anais...** [S.l.: s.n.], 2019. p.458–467.

CORRÊA, G.; ASSUNCAO, P.; AGOSTINI, L.; SILVA CRUZ, L. A. da. Complexity control of high efficiency video encoders for power-constrained devices. **IEEE Transactions on Consumer Electronics**, [S.l.], v.57, n.4, p.1866–1874, 2011.

CORRÊA, G.; ASSUNçãO, P.; AGOSTINI, L.; CRUZ, L. A. d. S. Coding Tree Depth Estimation for Complexity Reduction of HEVC. In: IEEE DATA COMPRESSION CONFERENCE (DCC), 2013. **Anais...** [S.l.: s.n.], 2013. p.43–52.

CORRÊA, G.; ASSUNçãO, P.; AGOSTINI, L.; CRUZ, L. A. d. S. Complexity scalability for real-time HEVC encoders. **Journal of Real-Time Image Processing**, [S.l.], v.12, p.107–122, 2014.

CORRÊA, G.; ASSUNçãO, P.; CRUZ, L. A. d. S.; AGOSTIN, L. Adaptive Coding Tree for Complexity Control of High Efficiency Video Encoders. In: IEEE PICTURE CODING SYMPOSIUM (PCS), 2012. **Anais...** [S.l.: s.n.], 2012. p.425–428.

CORRÊA, G.; ASSUNçãO, P.; CRUZ, L. A. d. S.; AGOSTINI, L. Encoding time control system for HEVC based on Rate-Distortion-Complexity analysis. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS), 2015. **Anais...** [S.l.: s.n.], 2015. p.1114–1117.

CORRÊA, G.; ASSUNçãO, P.; CRUZ, L. A. d. S.; AGOSTINI, L. Pareto-Based Method for High Efficiency Video Coding With Limited Encoding Time. **IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY**, [S.l.], v.26, n.9, p.1734–1745, 2016.

CORRÊA, G. R. **Computacional Complexity Reduction and Scaling for High Efficiency Video Encoders**. 2014. 286p. Tese — Faculdade de Ciências e Tecnologia, Universidade de Coimbra, Coimbra.

CORRÊA, M. M.; WASKOW, B. H.; GOEBEL, J. W.; PALOMINO, D. M. A High-Throughput Hardware Architecture for AV1 Non-Directional Intra Modes. In: IEEE DATA COMPRESSION CONFERENCE (DCC), 2020. **Anais...** [S.l.: s.n.], 2020. v.67, n.5, p.1481–1494.

DAEDE, T.; NORKIN, A.; BRAILOVSKIY, I. **Video Codec Testing and Quality Measurement draft-ietf-netvc-testing-09**. 2020. Disponível em: <https://datatracker.ietf.org/doc/html/draft-ietf-netvc-testing-09>. Acesso em: outubro de 2020.

DANGETI, P. **Statistics for Machine Learning**. [S.l.]: Packt Publishing, 2017.

DENG, X. et al. Subjective-Driven Complexity Control Approach for HEVC. **IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY**, [S.l.], v.26, n.1, p.91–106, 2016.

DENG, X.; XU, M.; LI, C. Hierarchical Complexity Control of HEVC for Live Video Encoding. **IEEE Access**, [S.l.], v.4, p.1014–1027, 2016.

DHAMDHERE, S. N. Cumulative citations index, h-index and i10-index (research metrics) of an educational institute: A case study. **International Journal of Library and Information Science**, [S.l.], v.10, n.1, p.1–9, 2018.

DORF, R. C.; BISHOP, R. H. **Modern Control Systems**. [S.l.]: Pearson Education, 2017.

FANG, J.-T.; TU, Y.-L.; YU, L.-P.; CHANG, P.-C. Real-time Complexity Control for High Efficiency Video Coding. In: IEEE INTERNATIONAL CONFERENCE ON INFORMATION COMMUNICATION AND SIGNAL PROCESSING (ICSP), 2018. **Anais...** [S.l.: s.n.], 2018. p.85–89.

FONSECA, J. **GitHub**. 2020. Disponível em: <https://github.com/jrfonseca/Gprof2dot>. Acesso em: julho de 2020.

GANKHUYAG, G.; JEONG, J.; KIM, Y. Advanced Motion-Constrained AV1 Encoder for 8K 360 VR Tiled Streaming. In: IEEE INTERNATIONAL CONFERENCE ON INFORMATION AND COMMUNICATION TECHNOLOGY CONVERGENCE (ICTC), 2019. **Anais...** [S.l.: s.n.], 2019. p.682–684.

GHANBARI, M. **Standard Codecs**: Image Compression to Advanced Video Coding. [S.l.]: United Kingdon: The Institution of Electrical Engineers and Technology, 2003.

GHOSH, A. Evolutionary Algorithms for Multi-Criterion Optimization: A Survey. **International Journal of Computing & Information Sciences**, [S.l.], v.2, n.1, p.38–57, 2004.

GITHUB. **M2cgen (Model 2 Code Generator)**. 2022. Disponível em: <https://github.com/BayesWitnesses/m2cgen>. Acesso em: junho de 2022.

GONZALEZ, R.; WOODS, R. **Processamento de Imagens Digitais**. [S.l.]: Blüchers, 2003.

GOOGLE. **Google Acadêmico**. 2022. Disponível em: <https://scholar.google.com.br/>. Acesso em: abril de 2022.

GPROF. 2009. Disponível em: <https://linux.die.net/man/1/gprof>. Acesso em: setembro de 2020.

GRELLERT, M. et al. AN ADAPTIVE WORKLOAD MANAGEMENT SCHEME FOR HEVC ENCODING. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2013. **Anais...** [S.l.: s.n.], 2013. p.1850–1854.

GRELLERT, M. et al. Complexity control of HEVC encoders targeting real-time constraints. **Journal of Real-Time Image Processing**, [S.l.], v.13, p.5–24, 2017.

GU, J.; WEN, J. Mid-Depth Based Blocks Structure Determination for AV1. In: IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING (ICASSP), 2019. **Anais...** [S.l.: s.n.], 2019. p.1617–1621.

GUO, B.; HAN, Y.; WEN, J. Fast Block Structure Determination in Av1-Based Multiple Resolutions Video Encoding. In: IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA AND EXPO (ICME), 2018. **Anais...** [S.l.: s.n.], 2018.

GUO, B. et al. A Bayesian Approach to Block Structure Inference in AV1-Based Multi-Rate Video Encoding. In: IEEE DATA COMPRESSION CONFERENCE (DCC), 2018. **Anais...** [S.l.: s.n.], 2018. p.383–392.

GUO, L. et al. **Netflix Now Streaming AV1 on Android**. 2020. Disponível em: <https://netflixtechblog.com/netflix-now-streaming-av1-on-android-d5264a515202>. Acesso em: março de 2020.

HAN, J.; KAMBER, M.; J., P. **Data Mining**: Concepts and Techniques. [S.l.]: Morgan Kaufmann Publishers, 2011.

HAN, J. et al. A Technical Overview of AV1. **Proceedings of the IEEE**, [S.l.], v.109, n.9, p.1435–1462, 2021.

HOSSEINI, E.; PAKDAMAN, F.; HASHEMI, M. R.; GHANBARI, M. Fine-grain complexity control of HEVC intra prediction in battery-powered video codecs. **Journal of Real-Time Image Processing**, [S.l.], v.18, p.603–618, 2020.

HSU, P.-K.; SHEN, C.-A. The VLSI Architecture of a Highly Efficient Deblocking Filter for HEVC Systems. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.27, n.5, p.1091–1103, 2017.

HUANG, C. et al. ENCODING COMPLEXITY CONTROL FOR LIVE VIDEO APPLICATIONS: AN INTERPRETABLE MACHINE LEARNING APPROACH. In: IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA AND EXPO (ICME), 2019. **Anais...** [S.l.: s.n.], 2019. p.1456–1461.

HUANG, C. et al. Online Learning-Based Multi-Stage Complexity Control for Live Video Coding. **IEEE TRANSACTIONS ON IMAGE PROCESSING**, [S.l.], v.30, p.641–656, 2021.

HUANG, Y. et al. Modeling Acceleration Properties for Flexible Intra HEVC Complexity Control. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.31, n.11, p.4454–4469, 2021a.

HUANG, Y. et al. Intra Encoding Complexity Control with a Time-Cost Model for Versatile Video Coding. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS), 2022. **Anais...** [S.l.: s.n.], 2022. p.2027–2031.

HUANG, Y. et al. Precise Encoding Complexity Control for Versatile Video Coding. **IEEE Transactions on Broadcasting**, [S.l.], p.1–16, 2022a.

HUNTER, J. D. Matplotlib: A 2D graphics environment. **Computing in science & engineering**, [S.l.], v.9, n.03, p.90–95, 2007.

HWANG, W.-L.; LEE, C.-C.; PENG, G.-J. Multi-Objective Optimization and Characterization of Pareto Points for Scalable Coding. **IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY,**, [S.l.], v.29, n.7, p.2096–2111, 2019.

INC., K. B. **Color Blindness Test-Ishihara**. 2022. Disponível em: <https://play.google.com/store/apps/dev?id=5997686785003835939&glŪS>. Acesso em: dezembro de 2022.

ISO/IEC. **Information technology – Coding of audio-visual objects – Part 10**: Advanced Video Coding. ISO/IEC 14496-10:2003. 2003. Disponível em: <https://www.iso.org/standard/37729.html>. Acesso em: outubro de 2020.

ISO/IEC. **Information technology – High efficiency coding and media delivery in heterogeneous environments – Part 2**: High efficiency video coding. ISO/IEC 23008-2. 2013. Disponível em: <https://www.iso.org/standard/35424.html>. Acesso em: outubro de 2020.

ITU-R. **Recommendation ITU-R BT.500-13 - Methodology for the subjective assessment of the quality of television pictures**. 2012. Disponível em: <https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.500-13-201201-I!!PDF-E.pdf>. Acesso em: outubro de 2021.

ITU-T. **SERIES P**: TELEPHONE TRANSMISSION QUALITY, TELEPHONE INSTALLATIONS, LOCAL LINE NETWORKS:Audiovisual quality in multimedia services - Subjective video quality assessment methods for multimedia applications

(P9.10). 1999. Disponível em: <https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-P.910-199909-S!!PDF-E&type=items>. Acesso em: outubro de 2021.

ITU-T. **H.264**: Advanced video coding for generic audiovisual services. Recommendation H.264 (05/03). 2003. Disponível em: <https://www.itu.int/rec/T-REC-H.264>. Acesso em: outubro de 2020.

ITU-T. **H.265**: High efficiency video coding. Recommendation H.265 (04/13). 2013. Disponível em: <https://www.itu.int/rec/T-REC-H.265>. Acesso em: outubro de 2020.

JEONG, J.; GANKHUYAG, G.; KIM, Y. A Fast Intra Mode Decision Based on Accuracy of Rate Distortion Model for AV1 Intra Encoding. In: IEEE INTERNATIONAL TECHNICAL CONFERENCE ON CIRCUITS / SYSTEMS, COMPUTERS AND COMMUNICATIONS (ITC-CSCC), 2019. **Anais...** [S.l.: s.n.], 2019.

JEONG, J.; GANKHUYAG, G.; KIM, Y. Fast Chroma Prediction Mode Decision based on Luma Prediction Mode for AV1 Intra Coding. In: IEEE INTERNATIONAL CONFERENCE ON INFORMATION AND COMMUNICATION TECHONOLOGY CONVERGENCE (ICTC), 2019a. **Anais...** [S.l.: s.n.], 2019a. p.1050–1052.

JIMENEZ-MORENO, A.; MARTíNEZ-ENRíQUEZ, E.; MARíA, F. Díaz-de. Complexity Control Based on a Fast Coding Unit Decision Method in the HEVC Video Coding Standard. **IEEE TRANSACTIONS ON MULTIMEDIA**, [S.l.], v.18, n.4, p.563–575, 2016.

JIN, Y. et al. IMPROVED INTRA MODE CODING BEYOND AV1. In: IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING (ICASSP), 2021. **Anais...** [S.l.: s.n.], 2021. p.1580–1584.

KIM, J.; BLASI, S.; DIAS A. S.AND MRAK, M.; IZQUIERDO, E. Fast Inter-Prediction Based on Decision Trees for AV1 Encoding. In: IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING (ICASSP), 2019. **Anais...** [S.l.: s.n.], 2019. p.1627–1631.

KUHN, P. **Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation**. [S.l.]: Springer New York, NY, 1999.

LAMORTE, W. W. **Characteristics of a Normal Distribution**. 2022. Disponível em: <https://sphweb.bumc.bu.edu/otlt/MPH-Modules/PH717-QuantCore/PH717-Module6-RandomError/PH717-Module6-RandomError5.html>. Acesso em: junho de 2022.

LAUDE, T. et al. A Comparison of JEM and AV1 with HEVC: Coding Tools, Coding Efficiency and Complexity. In: IEEE PICTURE CODING SYMPOSIUM (PCS), 2018. **Anais...** [S.l.: s.n.], 2018. p.36–40.

LI, T.; XU, M.; DENG, X.; SHEN, L. Accelerate CTU Partition to Real Time for HEVC Encoding With Complexity Control. **IEEE TRANSACTIONS ON IMAGE PROCESSING**, [S.l.], v.29, p.7482–7496, 2020.

LIN, W.-T. et al. Efficient AV1 Video Coding Using a Multi-layer Framework. In: IEEE DATA COMPRESSION CONFERENCE (DCC), 2018. **Anais...** [S.l.: s.n.], 2018. p.365–373.

MANSRI, I. et al. Comparative Evaluation of VVC, HEVC, H.264, AV1, and VP9 Encoders for Low-Delay Video Applications. In: IEEE FOURTH INTERNATIONAL CONFERENCE ON MULTIMEDIA COMPUTING, NETWORKING AND APPLICATIONS (MCNA), 2020. **Anais...** [S.l.: s.n.], 2020. p.38–43.

MCKINNEY, W. et al. pandas: a foundational Python library for data analysis and statistics. **Python for high performance and scientific computing**, [S.l.], v.14, n.9, p.1–9, 2011.

MICROSOFT. **LIGHTGBM documentation**. 2022. Disponível em: <https://lightgbm.readthedocs.io/en/latest/index.html>. Acesso em: junho de 2022.

MUKHERJEE, D. et al. A Technical Overview of VP9 – The Latest Open-Source Video Codec. In: IEEE ANNUAL TECHNICAL CONFERENCE & EXHIBITION (SMPTE), 2013. **Anais...** [S.l.: s.n.], 2013. p.54–44.

NASER, M. Z.; ALAVI, A. Insights into Performance Fitness and Error Metrics for Machine Learning. **CoRR**, [S.l.], v.abs/2006.00887, 2020.

NETINT. **NETINT Announces the World's First Commercially Available Hardware AV1 Encoder for the Data Center**. 2021. Disponível em: <https://netint.ca/netint-announces-the-worlds-first-commercially-available-hardware-av1-encoder-for-the-data-center/>. Acesso em: julho de 2021.

NGUYEN, T.; MARPE, D. Future Video Coding Technologies: A Performance Evaluation of AV1, JEM, VP9, and HM. In: IEEE PICTURE CODING SYMPOSIUM (PCS), 2018. **Anais...** [S.l.: s.n.], 2018. p.31–35.

NVIDIA. **AMPERE GA102 GPU ARCHITECTURE**. 2021. Disponível em: <https://www.nvidia.com/content/PDF/nvidia-ampere-ga-102-gpu-architecture-whitepaper-v2.pdf>. Acesso em: julho de 2021.

OPPENHEIM, A.; SCHAFER, R.; STOCKHAM, T. Nonlinear filtering of multiplied and convolved signals. **Proceedings of the IEEE**, [S.l.], v.56, n.8, p.1264 – 1291, 1968.

PAKDAMAN, F. et al. SVM based approach for complexity control of HEVC intra coding. **Signal Processing: Image Communication**, [S.l.], v.96, p.1–14, 2021.

PARKER, S. et al. On transform coding tools under development for VP10. In: APPLICATIONS OF DIGITAL IMAGE PROCESSING XXXIX, 2016. **Anais...** SPIE, 2016. v.9971, p.407 – 416.

PARKER, S. et al. Global and locally adaptive warped motion compensation in video compression. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2017. **Anais...** [S.l.: s.n.], 2017. p.275–279.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **the Journal of machine Learning research**, [S.l.], v.12, p.2825–2830, 2011.

PENNY, W. et al. Pareto-based energy control for the HEVC encoder. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2016. **Anais...** [S.l.: s.n.], 2016. p.814–818.

PORTO, M. S. **Desenvolvimento Algorítmico e Arquitetural para a Estimação de Movimento na Compressão de Vídeo de Alta Definição**. 2012. 166p. Tese — Universidade Federal do Rio Grande do Sul. Instituto de Informática. Programa de Pós-Graduação em Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre.

PURI, A.; CHEN, X.; LUTHRAC, A. Video Coding Using the H.264/MPEG-4 AVC Compression Standard. **Elsevier Signal Processing: Image Communication**, [S.l.], v.19, n.9, p.793–849, 2004.

RESEACH, G. V. **Video Streaming Market Size, Share & Trends Analysis Report By Streaming Type, By Solution, By Platform, By Service, By Revenue Model, By Deployment Type, By User, By Region, And Segment Forecasts, 2022 - 2030**. 2022. Disponível em: <https://www.grandviewresearch.com/industry-analysis /video-streaming-market#: :text=The%20global%20video% 20streaming%20market,used%20to%20improve%20video %20quality./>. Acesso em: outubro de 2022.

RICHARDSON, I. E. G. **Video Codec Design** : Developing Image and Video Compression. [S.l.]: John Wiley & Sons, 2002.

RIVAZ, P.; HAUGHTON, J. **AV1 Bitstream & Decoding Process Specification**. 2018. Disponível em: <https://aomedia.org/>. Acesso em: fevereiro de 2019.

SALDANHA, M. et al. An Overview of Dedicated Hardware Designs for State-of-the-Art AV1 and H.266/VVC Video Codecs. In: IEEE INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS AND SYSTEMS (ICECS), 2020. **Anais...** [S.l.: s.n.], 2020.

SAMALA, R. K.; CHAN, H.-P.; HADJIISKI, L.; KONERU, S. Hazards of data leakage in machine learning: a study on classification of breast cancer using deep neural networks. In: MEDICAL IMAGING 2020: COMPUTER-AIDED DIAGNOSIS, 2020. **Anais...** [S.l.: s.n.], 2020.

SIQUEIRA I.AND CORRÊA, G.; GRELLERT, M. Rate-Distortion and Complexity Comparison of HEVC and VVC Video Encoders. In: IEEE LATIN AMERICAN SYMPOSIUM ON CIRCUITS SYSTEMS (LASCAS), 2020. **Anais...** [S.l.: s.n.], 2020. p.1–4.

SMITH, S. W. **The scientist and engineer's guide to digital signal processing**. [S.l.]: California Technical Publishing, 1997.

SOARES, L. B.; DINIZ, C. M.; COSTA, E. A. C. da; BAMPI, S. In: IEEE SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN (SBCCI), 2018. **Anais...** [S.l.: s.n.], 2018.

SU, H. et al. Machine Learning Accelerated Transform Search for AV1. In: IEEE PICTURE CODING SYMPOSIUM (PCS), 2019. **Anais...** [S.l.: s.n.], 2019.

SULLIVAN, G. J.; OHM, J.-R.; HAN, W.-J.; WIEGAND, T. Overview of the High Efficiency Video Coding (HEVC) Standard. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.22, n.12, p.1649–1668, 2012.

WEBM. **About WebM**. 2012. Disponível em: <https://www.webmproject.org/about/>. Acesso em: fevereiro de 2019.

WEBSTER, A. A.; WOLF, S. **Spatial and Temporal Information Measures for Video Quality. ANSI T1Q1 Contribution T1Q1.5/92-113**. 1992. Disponível em: <https://datatracker.ietf.org/doc/html/draft-ietf-netvc-testing-09>. Acesso em: outubro de 2020.

XGBOOST. **XGBoost Python package**. 2022. Disponível em: <https://xgboost.readthedocs.io/en/stable/python/index.html>. Acesso em: junho de 2022.

XIPH. **rave1**: The fastest and safest AV1 encoder. 2020. Disponível em: <https://github.com/xiph/rav1e>. Acesso em: outubro de 2022.

XIPH. **AV2 Candidate Test Media**. 2022. Disponível em: <https://media.xiph.org/video/av2/>. Acesso em: abril de 2022.

XIPH. **Xiph.org Video Test Media [derf's collection]**. 2022. Disponível em: <https://media.xiph.org/video/derf/>. Acesso em: julho de 2022.

XU, M.; JEON, B. User-Priority Based AV1 Coding Tool Selection. **IEEE Transactions on Broadcasting**, [S.l.], v.67, n.3, p.736 – 745, 2021.

YOUTUBE. **YouTube for Press**. 2022. Disponível em: <https://blog.youtube/press/>. Acesso em: abril de 2022.

ZHANG, J.; KWONG, S. T. W.; ZHAO, T.; IP, H. H. S. Complexity Control in the HEVC Intracoding for Industrial Video Applications. **IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS**, [S.l.], v.15, n.3, p.1437–1449, 2019.

ZHANG, J.; KWONG, S.; ZHAO, T.; PAN, Z. CTU-Level Complexity Control for High Efficiency Video Coding. **IEEE TRANSACTIONS ON MULTIMEDIA**, [S.l.], v.20, n.1, p.29–44, 2018.

ZHANG, J. et al. Complexity Control for HEVC Inter Coding Based on Two-Level Complexity Allocation and Mode Sorting. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2018a. **Anais...** [S.l.: s.n.], 2018a. p.3628–3632.

Apêndices

APÊNDICE A – Parâmetros de codificação do codificador AV1

O AV1 possui diversos parâmetros de codificação, os quais contribuem para a complexidade do codificador. Devido a esta grande quantidade, a análise de todos os parâmetros existentes torna-se inviável. Recentemente, foi apresentado por Xu; Jeon (2021) uma análise de diversos parâmetros de codificação AV1. No entanto, não são exatamente os mesmos mostrados neste trabalho. Aqui é apresentada uma análise mais ampla dos parâmetros de predição inter-quadros, além de parâmetros relacionados ao estágio de transformadas e particionamento, os quais não são contemplados pelo artigo mencionado. Para o desenvolvimento deste trabalho, com base nos resultados apresentados na Seção 5.2, optou-se por investigar a sensibilidade do codificador AV1 em relação aos parâmetros relacionados à predição inter-quadros e transformadas. Esses estágios são os que mais consomem tempo de codificação Libaom, logo seus parâmetros, provavelmente, influenciam de maneira significativa na complexidade do codificador. Outros fortes candidatos são os parâmetros de particionamento, uma vez que a escolha do particionamento dos quadros afeta, simultaneamente, diferentes estágios de codificação. A seguir serão listados os 23 parâmetros analisados. Os primeiros treze parâmetros estão relacionados à predição inter-quadros, os próximos quatro a etapa de transformadas e os seis últimos estão relacionados ao particionamento. Mais detalhes relacionados aos parâmetros citados podem ser encontrados no Capítulo 3.

- **dist-wtd-comp:** habilita/desabilita a predição inter-quadros Compound Distance. É desabilitado em 0 e habilitado em 1 (padrão).
- **masked-comp:** habilita/desabilita o uso de máscaras, ao mesmo tempo, em quatro predições compostas, Interinter Wedge Compound, Difference-Weighted Compound, Inter-Intra Wedge Compound e Inter-Intra Compound. É desabilitado em 0 e habilitado em 1 (padrão).
- **onesided-comp:** habilita/desabilita one sided compound, o qual é usado apenas quando todos os quadros de referências são unilaterais, ou seja, são quadros passados ou quadros futuros. É desabilitado em 0 e habilitado em 1 (padrão).
- **interintra-comp:** habilita/desabilita o modo inter-quadros Compound Inter-Intra. É desabilitado em 0 e habilitado em 1 (padrão).

- **smooth-interintra** habilita/desabilita os modos smooth da predição Compound Inter-Intra. É desabilitado em 0 e habilitado em 1 (padrão).
- **diff-wtd-comp:** habilita/desabilita a predição inter-quadros Compound Diffwtd. É desabilitado em 0 e habilitado em 1 (padrão).
- **interinter-wedge:** habilita/desabilita a predição inter-quadros Compound Wedge. É desabilitado em 0 e habilitado em 1 (padrão).
- **interintra-wedge:** habilita/desabilita o modo wedge na predição Compound Inter-Intra. É desabilitado em 0 e habilitado em 1 (padrão).
- **global-motion:** habilita/desabilita o modo de predição Global Motion. É desabilitado em 0 e habilitado em 1 (padrão).
- **warped-motion:** habilita/desabilita o modo de predição Warped Motion. É desabilitado em 0 e habilitado em 1 (padrão).
- **obmc:** habilita/desabilita o modo de predição Overlapped Block Motion Compensation. É desabilitado em 0 e habilitado em 1 (padrão).
- **max-reference-frames** define o número máximo de quadros de referência. Pode usar de três a sete quadros de referência. A configuração padrão usa o número máximo de quadros permitido.
- **ref-frame-mvs:** habilita/desabilita o uso de vetores de movimento de outro quadro de referência para prever o vetor de movimento do bloco atual. É desabilitado em 0 e habilitado em 1 (padrão).
- **tx64:** habilita/desabilita tamanhos de blocos 64×64, 64×32, 32×64, 64×16 e 16×64. É desabilitado em 0 e habilitado em 1 (padrão).
- **flip-idtx:** habilita/desabilita a utilização dos seguintes tipos de transformadas: FLIPADST_DCT, DCT_FLIPADST, FLIPADST_FLIPADST, ADST_FLIPADST, FLIPADST_ADST, IDTX, V_DCT, H_DCT, V_ADST, H_ADST, V_FLIPADST, H_FLIPADST. É desabilitado em 0 e habilitado em 1 (padrão).
- **reduced-tx-type-set:** quando habilitado, igual = 1, limita o codificador ao conjunto INTER3 e INTRA2 da Tabela 3.
- **use-inter-dct-only:** habilitado/desabilita o uso apenas de transformada DCT na predição inter-quadros. É desabilitado em 0 e habilitado em 1 (padrão).
- **rect-partitions:** este parâmetro habilita/desabilita todas as partições retangulares. É desabilitado em 0 e habilitado em 1 (padrão).

- **ab-partitions:** parâmetro que habilita/desabilita partições do tipo AB. É desabilitado em 0 e habilitado em 1 (padrão).
- **1to4-partitions** parâmetro que habilita/desabilita partições do tipo 1:4 e 4:1. É desabilitado em 0 e habilitado em 1 (padrão).
- **min-partition-size:** define o tamanho mínimo de partição (4: 4×4, 8: 8×8, 16: 16×16, 32: 32×32, 64: 64×64, 128: 128×128).
- **max-partition-size:** define o tamanho máximo de partição (4: 4x4, 8: 8x8, 16: 16x16, 32: 32x32, 64: 64x64, 128: 128x128).
- **sb-size:** define o tamanho do superbloco (64×64 ou 128×128).

APÊNDICE B – Configurações analisadas

Parâmetros	Configurações																			
	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15	c16	c17	c18	c19
dist-wtd-comp	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
masked-comp	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
onesided-comp	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
interintra-comp	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
smooth-interintra	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
diff-wtd-comp	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
interinter-wedge	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
interintra-wedge	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
globol-motion	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
warped-motion	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
obmc	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
max-reference-frames	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
reference-frame-mvs	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
tx64	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
flip-idtx	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
reduced-tx-type-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
use-inter-dct-only	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rect-partitions	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ab-partitions	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1to4-partitions	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
min-partition-size	4	4	4	4	4	64	32	16	8	32	16	8	4	16	8	4	8	4	4	4
max-partition-size	128	128	128	128	128	128	128	128	128	64	64	64	64	32	32	32	16	16	8	128
sb-size	128	128	128	128	64	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128

Parâmetros	Configurações																		
	c20	**c21**	**c22**	**c23**	**c24**	**c25**	**c26**	**c27**	**c28**	**c29**	**c30**	**c31**	**c32**	**c33**	**c34**	**c35**	**c36**	**c37**	**c38**
dist-wtd-comp	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
masked-comp	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
onesided-comp	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
interintra-comp	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
smooth-interintra	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
diff-wtd-comp	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
interinter-wedge	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
interintra-wedge	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
globol-motion	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
warped-motion	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
obmc	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
max-reference-frames	7	7	7	7	7	7	7	7	7	7	3	4	5	6	7	7	7	7	7
reference-frame-mvs	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
tx64	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
flip-idtx	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
reduced-tx-type-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
use-inter-dct-only	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
rect-partitions	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ab-partitions	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1to4-partitions	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
min-partition-size	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
max-partition-size	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128
sb-size	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128

APÊNDICE C – Sequências de vídeos analisadas para observar a ocorrência dos tamanhos de GFGs

- BasketballDrive_1920x1080_50
- BQTerrace_1920x1080_60
- Cactus_1920x1080_50
- Kimono_1920x1080_24
- ParkScene_1920x1080_24
- aspen_1080p_60f
- ducks_take_off_1080p50_60f
- life_1080p30_60f
- Netflix_Aerial_1920x1080_60fps_8bit_420_60f
- Netflix_Boat_1920x1080_60fps_8bit_420_60f
- Netflix_FoodMarket_1920x1080_60fps_8bit_420_60f
- Netflix_PierSeaside_1920x1080_60fps_8bit_420_60f
- old_town_cross_1080p50_60f
- pan_hdr_amazon_1080p
- park_joy_1080p50_60f
- pedestrian_area_1080p25_60f
- rush_field_cuts_1080p_60f
- rush_hour_1080p25_60f
- seaplane_hdr_amazon_1080p
- station2_1080p25_60f

- touchdown_pass_1080p_60f
- Netflix_Dancers_4096x2160_60fps_10bit_420_60f
- Netflix_Narrator_4096x2160_60fps_10bit_420_60f
- Netflix_WindAndNature_4096x2160_60fps_10bit_420_60f

APÊNDICE D – Seleção dos Pontos de Controle

A Tabela 29 mostra os resultados médios obtidos através das cinco sequências HD 1080 e cinco UHD 4K, todas com 180 quadros, mencionadas no Capítulo 5. Os *CPs* listados constituem a frente de Pareto. Os que estão destacados foram selecionados, após ser aplicada a metodologia de seleção de Pontos de Controle, ilustrada na Figura 29. A seguir são vistos os *CPs* selecionados como Pontos de Controle e os critérios de seleção aplicados a cada um deles.

Tabela 29 – Pontos médios de controle.

CP	RTC (%)	BD-Rate (%)	Razão	D.P. da Razão
CP1	**0,05**	**1,20**	**15,20**	**8,87**
CP11	0,07	6,28	27,16	17,54
CP13	**0,11**	**8,55**	**30,94**	**15,96**
CP23	**0,20**	**16,36**	**41,55**	**23,80**
CP32	**0,43**	**27,34**	**49,83**	**36,25**
CP38	**3,60**	**43,51**	**15,27**	**9,41**
CP39	11,57	45,86	7,01	7,39
CP41	23,25	57,78	7,25	7,56
CP42	**23,58**	**59,89**	**6,31**	**7,17**
CP44	**65,30**	**79,38**	**3,03**	**2,37**

- **CP1**: apresenta valores de BD-Rate igual a zero;
- **CP13**: apresenta BD-Rate negativo e tem razão menor que 2% em relação a razão do CP11;
- **CP23**: mais que 3% de RTC em relação ao último ponto selecionado;
- **CP32**: mais que 3% RTC em relação ao último ponto selecionado;
- **CP38**: razão maior que 2% em relação *CP39*;
- **CP42**: menor desvio padrão em relação a *CP41*;
- **CP44**: um valor de BD-Rate negativo.

Nas Tabelas 30 a 39 é possível observar os *PCs* selecionados (além do *PC0*), os *CPs* e os resultados de RTC e BD-Rate para cada uma das sequências analisadas.

Tabela 30 – Pontos de Controle para a sequência *Netflix_Crosswalk*, resolução HD 1080.

Pontos de Controle	Conjunto de Parâmetros	RTC (%)	BD-Rate (%)
PC5	CP8	2,55	-0,14
PC7	CP13	10,02	-0,09
PC12	CP23	18,73	0,01
PC15	CP32	28,96	0,15
PC20	CP38	44,09	3,98
PC24	CP42	61,08	41,69
PC26	CP44	72,74	155,79

Tabela 31 – Pontos de Controle para a sequência *Crowd_Run*, resolução HD 1080.

Pontos de Controle	Conjunto de Parâmetros	RTC (%)	BD-Rate (%)
PC7	CP13	8,83	0,07
PC15	CP32	25,76	0,21
PC23	CP41	34,96	2,02
PC24	CP42	47,38	3,28
PC26	CP44	77,08	11,72

Tabela 32 – Pontos de Controle para a sequência *Guitar_Hdr_Amazon*, resolução HD 1080.

Pontos de Controle	Conjunto de Parâmetros	RTC (%)	BD-Rate (%)
PC1	CP1	0,84	0,00
PC7	CP13	8,85	0,07
PC5	CP8	45,72	0,09
PC11	CP18	50,85	1,34
PC17	CP34	55,44	4,19
PC16	CP33	64,72	13,07
PC21	CP39	68,35	16,13
PC24	CP42	76,59	52,75
PC26	CP44	81,49	169,80

Tabela 33 – Pontos de Controle para a sequência *Netflix_SquareAndTimelapse*, resolução HD 1080.

Pontos de Controle	Conjunto de Parâmetros	RTC (%)	BD-Rate (%)
PC12	CP23	15,93	0,01
PC15	CP32	25,75	0,10
PC20	CP38	41,95	4,47
PC23	CP41	50,95	7,38
PC24	CP42	54,06	8,42
PC26	CP44	76,99	31,07

Tabela 34 – Pontos de Controle para a sequência *Netflix_TunnelFlag*, resolução HD 1080.

Pontos de Controle	Conjunto de Parâmetros	RTC (%)	BD-Rate (%)
PC1	CP1	1,08	0,00
PC8	CP14	12,16	0,41
PC15	CP32	21,18	0,69
PC19	CP36	37,53	7,03
PC21	CP39	43,51	12,08
PC23	CP41	52,47	17,22
PC24	CP42	56,19	17,89
PC26	CP44	75,31	76,92

Tabela 35 – Pontos de Controle para a sequência *Netflix_BarScene*, resolução UHD 4K.

Pontos de Controle	Conjunto de Parâmetros	RTC (%)	BD-Rate (%)
PC2	CP2	2,71	-0,34
PC4	CP6	10,33	-0,29
PC9	CP15	13,80	0,10
PC14	CP30	30,68	0,18
PC13	CP28	34,96	0,48
PC20	CP38	48,76	2,59
PC22	CP40	59,89	7,72
PC25	CP43	68,80	20,40
PC26	CP44	84,58	21,12

Tabela 36 – Pontos de Controle para a sequência *Netflix_BoxingPractice*, resolução UHD 4K.

Pontos de Controle	Conjunto de Parâmetros	RTC (%)	BD-Rate (%)
PC6	CP10	5,59	0,05
PC12	CP23	17,77	0,12
PC15	CP32	28,95	0,25
PC10	CP16	54,18	0,84
PC24	CP42	64,68	34,66
PC26	CP44	81,94	124,17

Tabela 37 – Pontos de Controle para a sequência *Netflix_ToddlerFountain*, resolução UHD 4K.

Pontos de Controle	Conjunto de Parâmetros	RTC (%)	BD-Rate (%)
PC6	CP10	4,66	-0,02
PC11	CP18	8,87	0,03
PC8	CP14	12,31	0,04
PC12	CP23	19,44	0,21
PC18	CP35	24,72	0,27
PC15	CP32	30,98	0,38
PC20	CP38	49,34	1,21
PC24	CP42	59,15	2,45
PC26	CP44	85,29	17,37

Tabela 38 – Pontos de Controle para a sequência *Netflix_RitualDance*, resolução UHD 4K.

Pontos de Controle	Conjunto de Parâmetros	RTC (%)	BD-Rate (%)
PC1	CP1	1,25	0,00
PC8	CP14	8,05	0,12
PC12	CP23	15,55	0,46
PC14	CP30	19,35	0,63
PC15	CP32	26,57	0,92
PC20	CP38	41,37	3,34
PC23	CP41	61,38	52,36
PC26	CP44	73,93	182,55

Tabela 39 – Pontos de Controle para a sequência *street_hdr_amazon*, resolução UHD 4K.

Pontos de Controle	Conjunto de Parâmetros	RTC (%)	BD-Rate (%)
PC3	CP3	4,87	-0,02
PC12	CP23	19,27	0,11
PC15	CP32	28,59	0,41
PC19	CP36	38,96	1,62
PC20	CP38	46,36	2,51
PC24	CP42	58,14	4,94
PC25	CP43	67,90	21,02
PC26	CP44	84,46	22,49

APÊNDICE E – Pseudocódigo dos controladores

Início *Controlador*
01 Seja $Quadro_{(0)}$ o primeiro quadro da sequência e n o número de quadros;
02 Seja $quantPC$ a quantidade de Pontos de Controle;
03 $i \leftarrow 0$; $j \leftarrow 0$; $k \leftarrow 0$; $index \leftarrow 0$; $tam \leftarrow 16$
04 $T_{GO} \leftarrow 0$; $T_D \leftarrow 0$; $R_T \leftarrow 0$;
05 **enquanto** ($i < tam$) **faça**
06 **codifique** $Quadro_{(i)}$ **com** $PC_{(0)}$;
07 **se** (i - 1 ≥ 0) **então**
08 $T_{GO} \leftarrow T_{GO} + T_{Q(i)}$;
09 **fim-se**;
10 $i \leftarrow i + 1$;
11 **fim-enquanto**;
12 **enquanto** ($i < n$) **faça**
13 $T_D \leftarrow$ tempo desejado para o próximo grupo {valor externo}
14 $R_T = T_D / T_{GO}$;
15 $j \leftarrow 0$;
16 **enquanto** ($j < quantPC$)
17 **se** (complexidade de $PC_{(j)}$ em relação a $PC_{(index)} \leq R_T$) **então**
18 $index \leftarrow j$;
19 **vá para** 24;
20 **fim-se**;
21 $j \leftarrow j + 1$;
22 **fim-enquanto**;
23 $index \leftarrow quantPC - 1$ {nenhum PC selecionado, seleciona $PC_{(quantPC-1)}$ }
24 $T_{GO} \leftarrow 0$;
25 **enquanto** ($k < tam$) ∧ ($i < n$) **faça**
26 **codifique** $Quadro_{(i)}$ **com** $PC_{(index)}$;
27 $T_{GO} \leftarrow T_{GO} + T_{Q(i)}$;
28 $i \leftarrow i + 1$;
29 $k \leftarrow k + 1$;
30 **fim-enquanto**;
31 $k \leftarrow 0$;
32 **fim-enquanto**;
Fim *Controlador*

Figura 61 – Pseudocódigo do controlador *baseline*.

```
Início Controlador
01    Seja Quadro(0) o primeiro quadro da sequência e n o número de quadros;
02    Seja quantPC(0) a quantidade de Pontos de Controle para a Tabela 0;
03    Seja PC(0)(0) o primeiro Ponto de Controle para a Classe 0;
04    TGO← 0; TD ← 0; TDA ← 0; RT ← 0; FC ← 0; TGPAnterior ← 0; TGPAtual ← 0;
05    i ← 0; j ← 0; k ← 0; index ← 0; budget ← 0; classe ← 0; últimaClasse ← 0;
06    stats ← Ø; Classificação ← Ø; Predição ← Ø;
07    enquanto (i < n) faça                          {primeira passada do AV1/libaom}
08          stats ← dados da primeira passada de Quadro(i);
09          Classificação(i) ← classificação de Quadro(i) usando stats;
10          Predição(i) ← predição de tempo de codificação de Quadro(i) usando stats;
11          i ← i + 1;
12    fim-enquanto;
13    Predição ← CorrigePredições(Predição);          {remove valores discrepantes}
14    i ← 0;
15    enquanto (i < 16) faça                         {segunda passada do AV1/libaom}
16          codifique Quadro(i) com PC(0)(0);
17          se (i - 1 ≥ 0) então
18                TGO ← TGO + TO(i);
19          fim-se;
20          i ← i + 1;
21    fim-enquanto;
22    enquanto (i < n) faça
23          se (TD ≠ 0 ) então                       {primeiro grupo não tem budget}
24                budget ← budget + (TD – TGO);
25          fim-se;
26          restandoGFG ← número de quadros até atualização de GFG;
27          últimaClasse ← classe;
28          classe ← ObtemClassificação(i, restandoGFG, Classificação);
29          TGPAnterior ← ObtemPredição(i – 16);
30          TGPAtual ← ObtemPredição(i);
31          se (TGPAnterior = 0) ∨ (TGPAtual = 0) então
32                TGPAnterior ← TGO;
33                TGPAtual ← TGO;
34                FC ← 1;
35          senão
36                FC ← TGO / TGPAnterior;
37          fim-se;
38          TDA ← TD + budget;
39          se (TDA < 1 ) então                      {evita valores de tempo negativos}
40                TDA ← 1;
41          fim-se;
42          TD ← tempo desejado para o próximo grupo                {valor externo}
43          RTA ← TDA / (TGPAtual × FC);
44          j ← 0;
45          se (classe ≠ últimaClasse) então   {classes diferentes, correção necessária}
46                RTA ← RTA × (complex. de PC(classe)(index) em relação a PC(classe)(0));
47                index ← 0;
48          fim-se;
49          enquanto (j < quantPC(classe))                  {seleciona PC adequado}
50                se (complex. de PC(classe)(j) em relação a PC(classe)(index) ≤ RTA) então
51                      index ← j;
52                      vá para 57;
53                fim-se;
54                j ← j + 1;
55          fim-enquanto;
56          index ← quantPC(classe) – 1;   {nenhum PC selecionado, seleciona PC(quantPC-1)}
57          TGQ ← 0;
58          enquanto (k < 16) ∧ (i < n) faça
59                codifique Quadro(i) com PC(index);
60                TGO ← TGO + TO(i) ;
61                i ← i + 1;
62                k ← k + 1;
63          fim-enquanto;
64          k ← 0;
65    fim-enquanto;
Fim Controlador
```

Figura 62 – Pseudocódigo do controlador *CCAM*.

```
Procedimento ObtemClassificação (index, restandoGFG, Classe)
01    Seja n o número de quadros;
02    Ocorr ← Ø; P ← Ø; i ← 0; próxQuadro ← 0; c ← 0;
03    P ← GeraPesos(restandoGFG);          {ponderação de cada quadro considerando GFG}
04    enquanto (i < 16) faça
05          próxQuadro ← index + i;
06          se (próxQuadro < n) então
07                c ← Classe(próxQuadro);
08                Ocorr (c) ← Ocorr (c) + P(i);          {soma com peso para o quadro i}
09          fim-se;
10          i ← i + 1;
11    fim-enquanto;
12    result ← índice onde houve maior número de ocorrências em Ocorr;
13    Retorne result;
Fim ObtemClassificação

Procedimento GeraPesos(restandoGFG)
01    Valores ← [10, 1, 2, 1, 4, 1, 2, 1, 6, 1, 2, 1, 3, 1, 2, 2];
02    i ← 0; Pesos ← Ø; index ← 0;
03    enquanto (i < 16) faça
04          index ← (i – restandoGFG + 16) mod 16;
05          Pesos(i) ← Valores(index);
06    fim-enquanto;
07    Retorne Pesos;
Fim GeraPesos

Procedimento ObtemPredição(quadro, Predição)
01    Seja n o número de quadros;
02    total ← 0;
03    se ((n – quadro) ≥ 16) então
04          total ← somatório de Predição(quadro) até Predição(quadro + 16);
05    fim-se;
06    Retorne total;
Fim ObtemPredição

Procedimento CorrigePredições(Predição)
01    Seja n o número de quadros;
02    média ← 0; dp ← 0; min ← 0; inf ← 0; sup ← 0; i ← 0;
03    min ← valor mínimo em Predição;
04    se (min < 0) então
05          adicione (–3 × min) a cada valor de Predição;
06    fim-se;
07    média ← média aritmética de Predição;
08    dp ← desvio padrão de Predição;
09    sup ← média + (3 × dp);
10    inf ← média – (3 × dp);
11    enquanto (i < n) faça
12          se (Predição(i) < inf) ∨ (Predição(i) > sup) então
13                Predição(i) ← média;
14          fim-se;
15    fim-enquanto;
16    Retorne Predição;
Fim CorrigePredições
```

Figura 63 – Continuação do pseudocódigo do controlador *CCAM*.

APÊNDICE F – Tabelas de transição usada em cada classe

Tabela 40 – *Netflix_Crosswalk*

Ponto de controle		**Anterior**							
		PC0	**PC5**	**PC7**	**PC12**	**PC15**	**PC20**	**PC24**	**PC26**
Próximo	PC0	**1,00**	1,03	1,11	1,23	1,41	1,79	2,57	3,67
	PC5	0,97	**1,00**	1,08	1,20	1,37	1,74	2,50	3,57
	PC7	0,90	0,92	**1,00**	1,11	1,27	1,61	2,31	3,30
	PC12	0,81	0,83	0,90	**1,00**	1,14	1,45	2,09	2,98
	PC15	0,71	0,73	0,79	0,87	**1,00**	1,27	1,83	2,61
	PC20	0,56	0,57	0,62	0,69	0,79	**1,00**	1,44	2,05
	PC24	0,39	0,40	0,43	0,48	0,55	0,70	**1,00**	1,43
	PC26	0,27	0,28	0,30	0,34	0,38	0,49	0,70	**1,00**

Tabela 41 – *Crowd_Run*

Ponto de controle		**Anterior**					
		PC0	**PC7**	**PC15**	**PC23**	**PC24**	**PC26**
Próximo	PC0	**1,00**	1,10	1,35	1,54	1,90	4,36
	PC7	0,91	**1,00**	1,23	1,40	1,73	3,98
	PC15	0,74	0,81	**1,00**	1,14	1,41	3,24
	PC23	0,65	0,71	0,88	**1,00**	1,24	2,84
	PC24	0,53	0,58	0,71	0,81	**1,00**	2,30
	PC26	0,23	0,25	0,31	0,35	0,44	**1,00**

Tabela 42 – *Guitar_Hdr_Amazon*

Ponto de controle		**Anterior**									
		PC0	**PC1**	**PC7**	**PC5**	**PC11**	**PC17**	**PC16**	**PC21**	**PC24**	**PC26**
Próximo	PC0	**1,00**	1,01	1,10	1,84	2,03	2,24	2,83	3,16	4,27	5,40
	PC1	0,99	**1,00**	1,09	1,83	2,02	2,23	2,81	3,13	4,24	5,36
	PC7	0,91	0,92	**1,00**	1,68	1,85	2,05	2,58	2,88	3,89	4,92
	PC5	0,54	0,55	0,60	**1,00**	1,10	1,22	1,54	1,72	2,32	2,93
	PC11	0,49	0,50	0,54	0,91	**1,00**	1,10	1,39	1,55	2,10	2,66
	PC17	0,45	0,45	0,49	0,82	0,91	**1,00**	1,26	1,41	1,90	2,41
	PC16	0,35	0,36	0,39	0,65	0,72	0,79	**1,00**	1,11	1,51	1,91
	PC21	0,32	0,32	0,35	0,58	0,64	0,71	0,90	**1,00**	1,35	1,71
	PC24	0,23	0,24	0,26	0,43	0,48	0,53	0,66	0,74	**1,00**	1,26
	PC26	0,19	0,19	0,20	0,34	0,38	0,42	0,52	0,58	0,79	**1,00**

Tabela 43 – *Netflix_SquareAndTimelapse*

Ponto de controle		Anterior						
		PC0	**PC12**	**PC15**	**PC20**	**PC23**	**PC24**	**PC26**
Próximo	PC0	**1,00**	1,19	1,35	1,72	2,04	2,18	4,35
	PC12	0,84	**1,00**	1,13	1,45	1,71	1,83	3,65
	PC15	0,74	0,88	**1,00**	1,28	1,51	1,62	3,23
	PC20	0,58	0,69	0,78	**1,00**	1,18	1,26	2,52
	PC23	0,49	0,58	0,66	0,85	**1,00**	1,07	2,13
	PC24	0,46	0,55	0,62	0,79	0,94	**1,00**	2,00
	PC26	0,23	0,27	0,31	0,40	0,47	0,50	**1,00**

Tabela 44 – *Netflix_TunnelFlag*

Ponto de controle		Anterior								
		PC0	**PC1**	**PC8**	**PC15**	**PC19**	**PC21**	**PC23**	**PC24**	**PC26**
Próximo	PC0	**1,00**	1,01	1,14	1,27	1,60	1,77	2,10	2,28	4,05
	PC1	0,99	**1,00**	1,13	1,26	1,58	1,75	2,08	2,26	4,01
	PC8	0,88	0,89	**1,00**	1,11	1,41	1,56	1,85	2,01	3,56
	PC15	0,79	0,80	0,90	**1,00**	1,26	1,40	1,66	1,80	3,19
	PC19	0,62	0,63	0,71	0,79	**1,00**	1,11	1,31	1,43	2,53
	PC21	0,56	0,57	0,64	0,72	0,90	**1,00**	1,19	1,29	2,29
	PC23	0,48	0,48	0,54	0,60	0,76	0,84	**1,00**	1,08	1,93
	PC24	0,44	0,44	0,50	0,56	0,70	0,78	0,92	**1,00**	1,77
	PC26	0,25	0,25	0,28	0,31	0,40	0,44	0,52	0,56	**1,00**

Tabela 45 – *Netflix_BarScene*

Ponto de controle		Anterior									
		PC0	**PC2**	**PC4**	**PC9**	**PC14**	**PC13**	**PC20**	**PC22**	**PC25**	**PC26**
Próximo	PC0	**1,00**	1,03	1,12	1,14	1,44	1,54	1,95	2,49	3,21	6,48
	PC2	0,97	**1,00**	1,09	1,11	1,40	1,50	1,90	2,43	3,12	6,31
	PC4	0,90	0,92	**1,00**	1,02	1,29	1,38	1,75	2,24	2,87	5,81
	PC9	0,88	0,90	0,98	**1,00**	1,26	1,35	1,71	2,18	2,81	5,68
	PC14	0,69	0,71	0,77	0,79	**1,00**	1,07	1,35	1,73	2,22	4,49
	PC13	0,65	0,67	0,73	0,74	0,94	**1,00**	1,27	1,62	2,08	4,22
	PC20	0,51	0,53	0,57	0,58	0,74	0,79	**1,00**	1,28	1,64	3,32
	PC22	0,40	0,41	0,45	0,46	0,58	0,62	0,78	**1,00**	1,29	2,60
	PC25	0,31	0,32	0,35	0,36	0,45	0,48	0,61	0,78	**1,00**	2,02
	PC26	0,15	0,16	0,17	0,18	0,22	0,24	0,30	0,38	0,49	**1,00**

Tabela 46 – *Netflix_BoxingPractice*

Ponto de controle		Anterior						
		PC0	**PC6**	**PC12**	**PC15**	**PC10**	**PC24**	**PC26**
Próximo	PC0	**1,00**	1,06	1,22	1,41	2,18	2,83	5,54
	PC6	0,94	**1,00**	1,15	1,33	2,06	2,67	5,23
	PC12	0,82	0,87	**1,00**	1,16	1,79	2,33	4,55
	PC15	0,71	0,75	0,86	**1,00**	1,55	2,01	3,93
	PC10	0,46	0,49	0,56	0,64	**1,00**	1,30	2,54
	PC24	0,35	0,37	0,43	0,50	0,77	**1,00**	1,96
	PC26	0,18	0,19	0,22	0,25	0,39	0,51	**1,00**

Tabela 47 – *Netflix_ToddlerFountain*

Ponto de controle		Anterior									
		PC0	**PC6**	**PC11**	**PC8**	**PC12**	**PC18**	**PC15**	**PC20**	**PC24**	**PC26**
Próximo	PC0	**1,00**	1,05	1,10	1,14	1,24	1,33	1,45	1,97	2,45	6,80
	PC6	0,95	**1,00**	1,05	1,09	1,18	1,27	1,38	1,88	2,33	6,48
	PC11	0,91	0,96	**1,00**	1,04	1,13	1,21	1,32	1,80	2,23	6,20
	PC8	0,88	0,92	0,96	**1,00**	1,09	1,16	1,27	1,73	2,15	5,96
	PC12	0,81	0,85	0,88	0,92	**1,00**	1,07	1,17	1,59	1,97	5,48
	PC18	0,75	0,79	0,83	0,86	0,93	**1,00**	1,09	1,49	1,84	5,12
	PC15	0,69	0,72	0,76	0,79	0,86	0,92	**1,00**	1,36	1,69	4,69
	PC20	0,51	0,53	0,56	0,58	0,63	0,67	0,73	**1,00**	1,24	3,44
	PC24	0,41	0,43	0,45	0,47	0,51	0,54	0,59	0,81	**1,00**	2,78
	PC26	0,15	0,15	0,16	0,17	0,18	0,20	0,21	0,29	0,36	**1,00**

Tabela 48 – *Netflix_RitualDance*

Ponto de controle		Anterior								
		PC0	**PC1**	**PC8**	**PC12**	**PC14**	**PC15**	**PC20**	**PC23**	**PC26**
Próximo	PC0	**1,00**	1,01	1,09	1,18	1,24	1,36	1,71	2,59	3,84
	PC1	0,99	**1,00**	1,07	1,17	1,22	1,34	1,68	2,56	3,79
	PC8	0,92	0,93	**1,00**	1,09	1,14	1,25	1,57	2,38	3,53
	PC12	0,84	0,86	0,92	**1,00**	1,05	1,15	1,44	2,19	3,24
	PC14	0,81	0,82	0,88	0,95	**1,00**	1,10	1,38	2,09	3,09
	PC15	0,73	0,74	0,80	0,87	0,91	**1,00**	1,25	1,90	2,82
	PC20	0,59	0,59	0,64	0,69	0,73	0,80	**1,00**	1,52	2,25
	PC23	0,39	0,39	0,42	0,46	0,48	0,53	0,66	**1,00**	1,48
	PC26	0,26	0,26	0,28	0,31	0,32	0,36	0,44	0,68	**1,00**

Tabela 49 – *street_hdr_amazon*

Ponto de controle		Anterior								
		PC0	**PC3**	**PC12**	**PC15**	**PC19**	**PC20**	**PC24**	**PC25**	**PC26**
Próximo	PC0	**1,00**	1,05	1,24	1,40	1,64	1,86	2,39	3,12	6,44
	PC3	0,95	**1,00**	1,18	1,33	1,56	1,77	2,27	2,96	6,12
	PC12	0,81	0,85	**1,00**	1,13	1,32	1,51	1,93	2,51	5,19
	PC15	0,71	0,75	0,88	**1,00**	1,17	1,33	1,71	2,22	4,60
	PC19	0,61	0,64	0,76	0,85	**1,00**	1,14	1,46	1,90	3,93
	PC20	0,54	0,56	0,66	0,75	0,88	**1,00**	1,28	1,67	3,45
	PC24	0,42	0,44	0,52	0,59	0,69	0,78	**1,00**	1,30	2,69
	PC25	0,32	0,34	0,40	0,45	0,53	0,60	0,77	**1,00**	2,07
	PC26	0,16	0,16	0,19	0,22	0,25	0,29	0,37	0,48	**1,00**

APÊNDICE G – Sequências utilizadas para coleta de dados para o modelo de preditor de tempo de codificação

Vídeos em 1080p:

- BasketballDrive_1920x1080
- blue_sky_1080p
- BQTerrace_1920x1080
- Cactus_1920x1080
- crowd_run_1080p50
- dinner_1080p30
- factory_1080p30
- FountainSky_1920x1080p30
- guitar_hdr_amazon_1080p
- in_to_tree_1080p50
- Kimono1_1920x1080_24
- Netflix_Aerial_1920x1080_60fps_8bit_420_60f
- Netflix_Boat_1920x1080_60fps_8bit_420_60f
- Netflix_Crosswalk_1920x1080_60fps_8bit_420_60f
- aspen_1080p_60f
- Netflix_SquareAndTimelapse_1920x1080_60fps_8bit_420_60f
- Netflix_TunnelFlag_1920x1080_60fps_8bit_420_60f
- old_town_cross_1080p
- pan_hdr_amazon_1080p
- parkscene 1080p24
- pedestrian_area_1080p25
- riverbed_1080p25
- rush_field_cuts_1080p
- life_1080p30_60f
- seaplane_hdr_amazon_1080p
- sintel_trailer_2k_1080p24
- station2_1080p25
- sunflower_1080p25
- touchdown_pass_1080p
- tractor_1080p25.y4m

- Wheat_1920x1080
- WorldCup_1920x1080_30p
- WorldCup_far_1920x1080_30p
- WorldCupFarSky_1920x1080_30p

Vídeos em 4K:

- cosmos_aom_1573-1749
- cosmos_aom_8686-8826
- cosmos_aom_11589-11752
- cosmos_aom_12149-12330
- cosmos_aom_12916-13078
- cosmos_aom_9561-9789
- cosmos_aom_13446-13649
- FlowerSky_B_3840x2160p30
- meridian_aom_1782-2163
- meridian_aom_11872-12263
- meridian_aom_12264-12745
- meridian_aom_15932-16309
- meridian_aom_20988-21412
- meridian_aom_22412-22738
- meridian_aom_24058-24550
- nocturne_aom_2370-2539
- nocturne_aom_8540-9009
- nocturne_aom_9010-9349
- nocturne_aom_17140-17709
- nocturne_aom_18013-18315
- nocturne_aom_23820-24322
- nocturne_aom_27740-28109
- nocturne_aom_32660-32799
- Netflix_BarScene_4096x2160_60fps
- Netflix_RitualDance_4096x2160_60fps
- Netflix_ToddlerFountain_4096x2160_60fps
- sol_levante_aom_289-453
- sol_levante_aom_3282-3874
- sol_levante_aom_4123-4545
- sparks_aom_30-511
- sparks_aom_2024-2455
- sparks_aom_5363-5763
- sparks_aom_5764-6024
- sparks_aom_8396-8941

- sparks_aom_11198-11570
- street_hdr_amazon_2160p

APÊNDICE H – Variáveis da primeira passada do Libaom analisadas para serem usadas como atributos para os modelos aprendizado de máquina

A seguir, são listadas e descritas as variáveis, da primeira passada do Libaom, analisadas para serem usadas como atributo nos *datasets* dos modelos de aprendizado de máquina utilizados pelo controlador *CCAM*. Essas variáveis foram coletadas das estruturas FIRSTPASS_STATS (fps) e FRAME_STATS (stats).

- **fps.weight:** Peso atribuído a um quadro com base no fator intra e no fator de brilho
- **fps.coded_error:** Melhor do erro de predição intra e erro de predição inter usando o *last frame* como referência
- **fps.sr_coded_error:** Melhor do erro de predição intra e erro de predição inter usando o quadro *golden* como referência
- **fps.tr_coded_error:** Melhor do erro de predição intra e erro de predição inter usando o quadro *altref* como referência
- **fps.intra_error:** Erro de predição intra
- **fps.frame_avg_wavelet_energy:** Energia wavelet média calculada usando Transformada Wavelet Discreta (DWT).
- **fps.count:** Número de quadros em uma coleção
- **fps.pcnt_inter:** Porcentagem de blocos com erro de predição inter < erro de predição intra
- **fps.pcnt_second_ref:** Porcentagem de blocos onde o quadro *golden* foi melhor que *last* ou intra
- **fps.pcnt_third_ref:** Porcentagem de blocos onde o quadro *altref* for melhor que intra, *last*, *golden*
- **fps.pcnt_neutral:** Porcentagem de blocos onde o erro de predição intra e inter foram bem próximos
- **fps.intra_skip_pct:** Porcentagem de blocos onde quase não há erro residual
- **fps.inactive_zone_rows:** Linhas superiores e inferiores mascaradas da imagem
- **fps.inactive_zone_cols:** Linhas da esquerda e direita mascaradas da imagem
- **fps.raw_error_stdev:** Desvio padrão do erro de predição de movimento
- **fps.MVr:** Média de vetores de movimento de linha
- **fps.mvr_abs:** Média dos valores absolutos de vetores de movimento de linhas

- **fps.MVc:** Média de vetores de movimento de colunas
- **fps.mvc_abs:** Média dos valores absolutos de vetores de movimento de colunas
- **fps.MVrv:** Variação de vetores de movimento de linha
- **fps.MVcv:** Variação dos vetores de movimento da coluna
- **fps.mv_in_out_count:** Valor que indica a fração de vetores de movimento de linha e coluna que são positivos ou negativos
- **fps.new_mv_count:** Contagem de vetores de movimento diferentes de zero únicos
- **fps.pcnt_motion:** Porcentagem de blocos usando previsão inter e vetores de movimento diferente de zero
- **stats.brightness_factor:** Fator de briho
- **stats.frame_avg_wavelet_energy:** Energia wavelet média calculada usando Transformada Wavelet Discreta (DWT)
- **stats.inter_count:** Contagem de blocos em que foi escolhido predição inter
- **stats.intra_factor:** Fator calculado usando o erro de predição intra
- **stats.intra_skip_count:** Contagem de blocos onde o erro intra é muito pequeno
- **stats.mv_count:** Contagem de vetor de movimento
- **stats.neutral_count:** Contagem de blocos onde o erro inter e o intra estão muito próximos e baixos
- **stats.new_mv_count:** Contagem de vetores de movimento diferentes de zero únicos
- **stats.second_ref_count:** Contagem de blocos que escolhem a segunda referência (*golden frame*)
- **stats.sr_coded_error:** Melhor do erro de predição intra e erro de predição inter usando o quadro *golden* como referência
- **stats.sum_in_vectors:** Soma de vetores de movimento para dentro
- **stats.sum_mvc:** Soma da coluna de vetores de movimento
- **stats.sum_mvc_abs:** Soma do valor absoluto da coluna do vetores de movimento
- **stats.sum_mvcs:** Soma do quadrado da coluna do vetores de movimento
- **stats.sum_mvr:** Soma da linha do vetores de movimento
- **stats.sum_mvr_abs:** Soma do valor absoluto da linha do vetores de movimento
- **stats.sum_mvrs:** Soma do quadrado da linha do vetores de movimento
- **stats.third_ref_count:** Contagem de blocos que escolhem a terceira referência (*altref frame*)

A seguir são listados os atributos usados para o treinamento do modelo predição de tempo de codificação e classificação.

Modelo de predição do tempo de codificação:

- fps.weight

- fps.coded_error
- fps.sr_coded_error
- fps.intra_error
- fps.frame_avg_wavelet_energy
- fps.pcnt_inter
- fps.pcnt_third_ref
- fps.pcnt_neutral
- fps.intra_skip_pct
- fps.inactive_zone_rows
- fps.raw_error_stdev
- fps.MVr
- fps.mvr_abs
- fps.MVc
- fps.mvc_abs
- fps.MVcv
- fps.mv_in_out_count
- fps.new_mv_count
- fps.pcnt_motion
- stats.brightness_factor
- stats.inter_count
- stats.mv_count
- stats.sr_coded_error
- stats.sum_mvc_abs
- CQ

Modelo de classificação:

- fps.weight
- fps.coded_error
- fps.sr_coded_error
- fps.intra_error
- fps.frame_avg_wavelet_energy
- fps.pcnt_inter
- fps.pcnt_third_ref
- fps.pcnt_neutral
- fps.intra_skip_pct
- fps.raw_error_stdev
- fps.MVr
- fps.mvr_abs
- fps.MVc
- fps.mvc_abs

- fps.MVcv
- fps.mv_in_out_count
- fps.new_mv_count
- fps.pcnt_motion
- stats.brightness_factor
- stats.inter_count
- stats.mv_count
- stats.sr_coded_error
- stats.sum_in_vectors
- stats.sum_mvc_abs
- stats.sum_mvrs

APÊNDICE I – Resultados obtidos para as sequências de vídeo usado no teste do modelo final

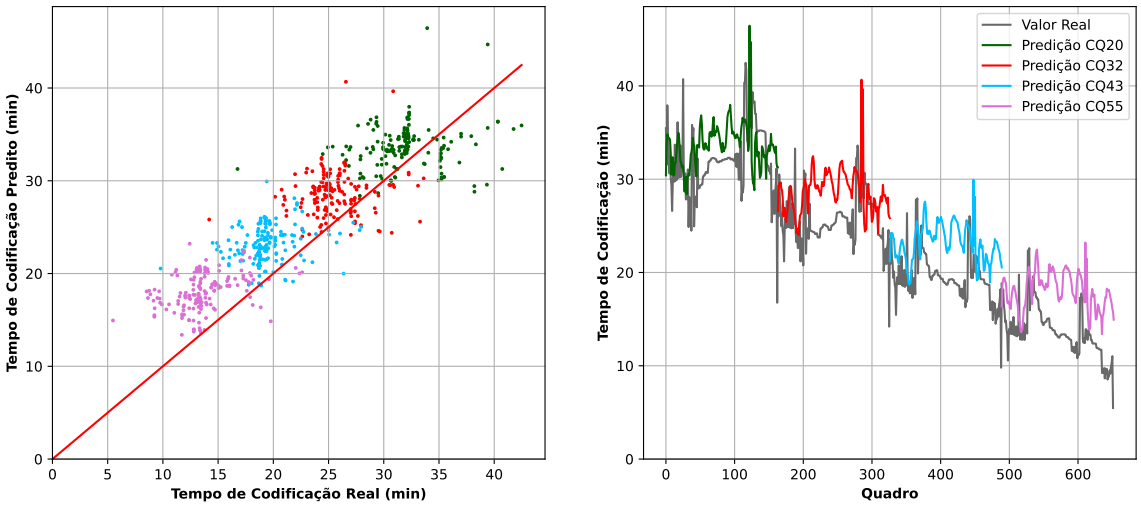


Figura 64 – Resultados de predição da sequência *BasketballDrive_1920x1080*.

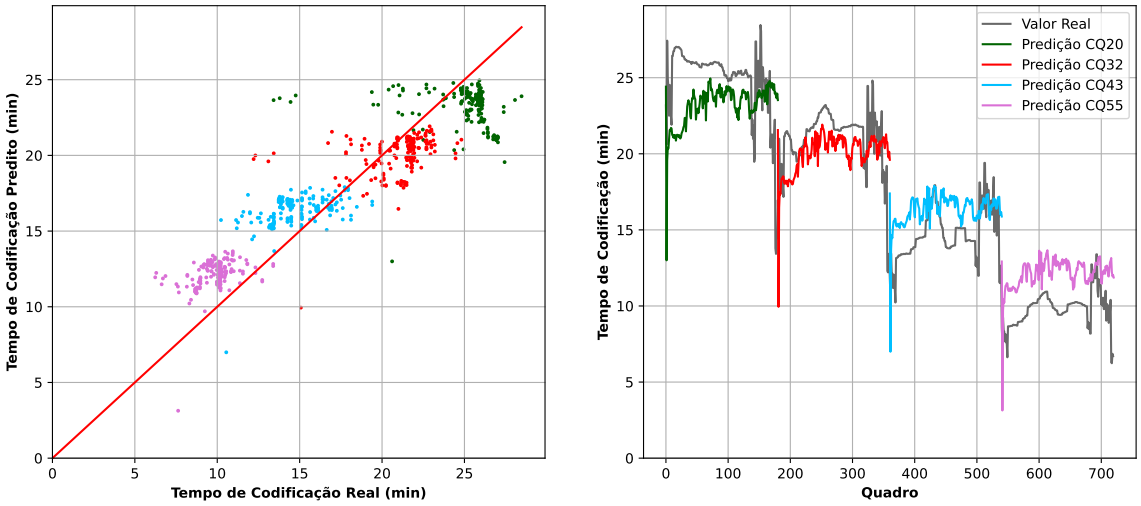


Figura 65 – Resultados de predição da sequência *In_to_tree_1920x1080*.

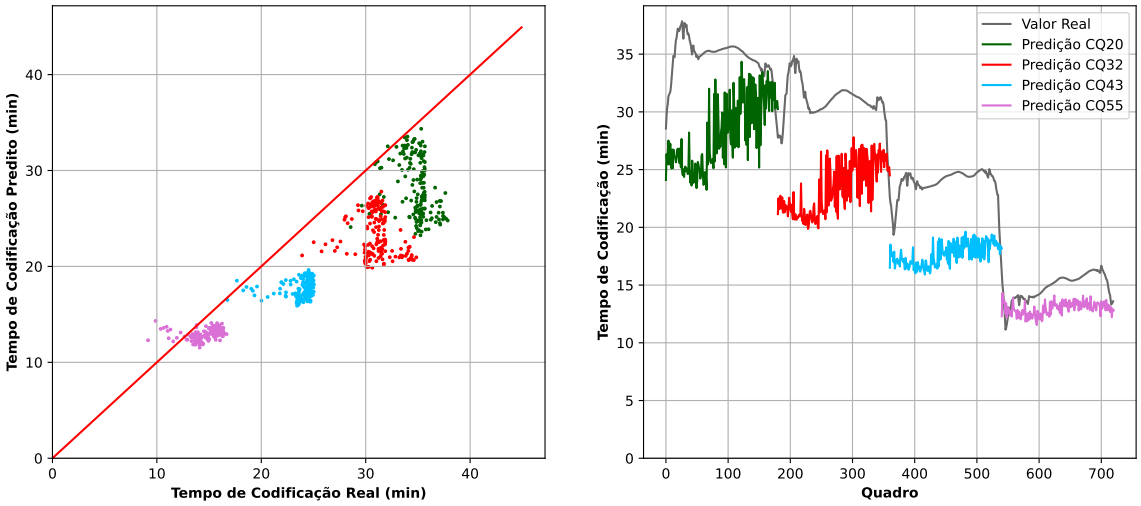


Figura 66 – Resultados de predição da sequência *ParkScene_1920x1080*.

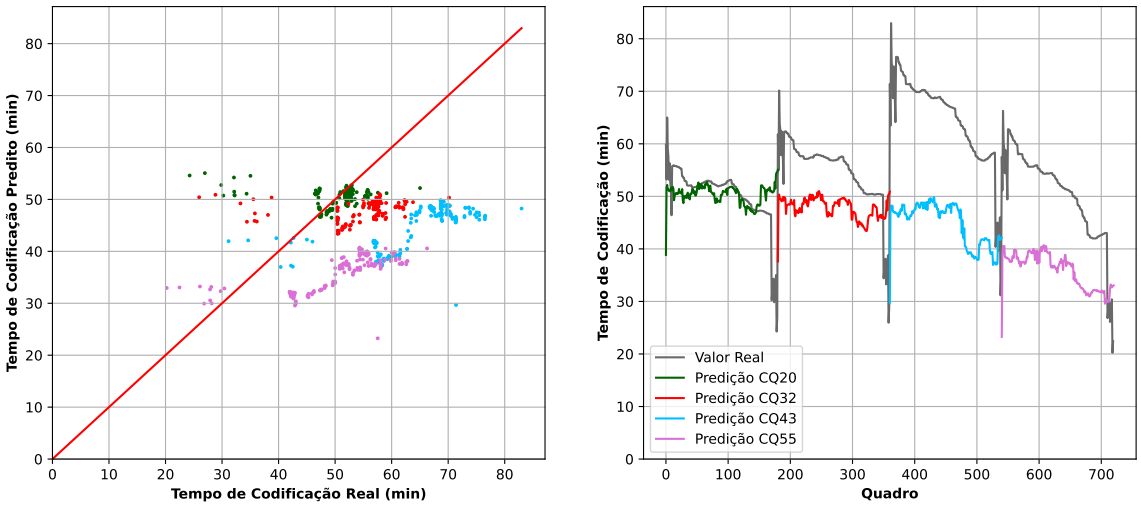


Figura 67 – Resultados de predição da sequência *Cosmos_aom_12916-13078*.

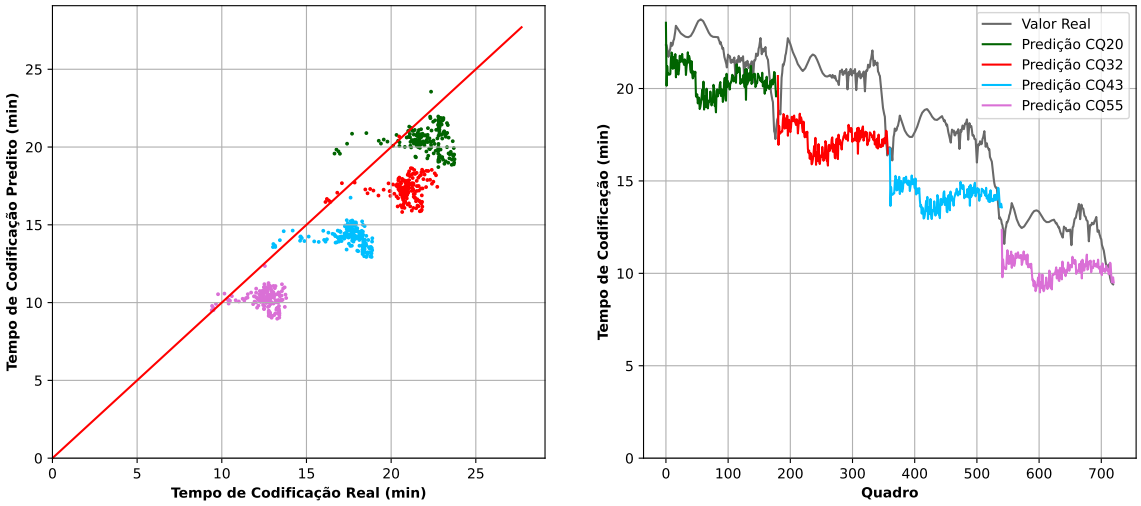


Figura 68 – Resultados de predição da sequência *Meridian_aom_sdr_12264-12745*.

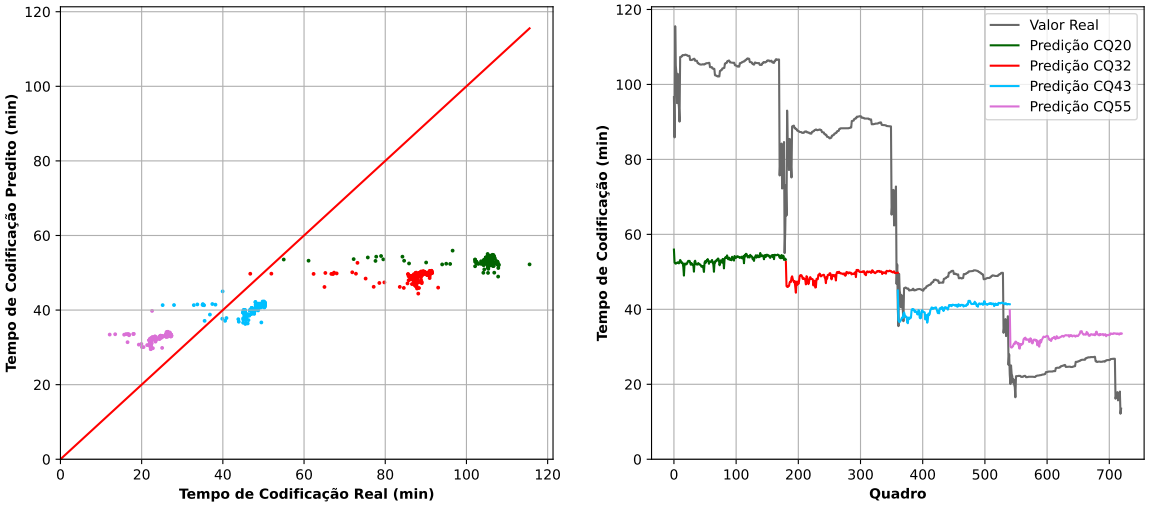


Figura 69 – Resultados de predição da sequência *Nocturne_aom_sdr_27740-28109*.

APÊNDICE J – Métodos de classificação analisados

Além do modo que utiliza a ponderação (número de ocorrência ponderada), mencionado na Seção 6.4, outras três estratégias foram analisadas com o objetivo de classificar os vídeos de entrada a cada grupo de 16 quadros, são elas: número de ocorrência, média e SI-TI. Assim como o modo "número de ocorrência ponderada" os modos "número de ocorrência" e "média" realiza a classificação do grupo de quadros a partir dos resultados obtidos para cada quadro com a utilização de aprendizado de máquina. No modo "número de ocorrência" a classe mais presente entre os 16 quadros é selecionada. Na média, a classe do grupo é definida a partir da média dos dados estatísticos dos 16 quadros.

Entretanto, o modo SI-TI não utiliza aprendizado de máquina. Nele, antes do processo de codificação, cada sequência de vídeo é classificada através do uso exclusivo das informações espaciais e temporais. A escolha da classe ocorre partir do cálculo da hipotenusa, obtida através da diferença dos valores SI e TI do vídeo de entrada em relação aos vídeos usados na análise de complexidade, Capítulo 5. Quanto menor o valor da hipotenusa, maior a semelhança entre os vídeos. Dessa forma, o vídeo de entrada é atribuído a classe respectiva ao vídeo com a qual obteve menor valor de hipotenusa.

Para considerar tanto a precisão quanto a qualidade promovida por cada uma das soluções analisadas, foi realizada uma análise com base na Equação 17. O custo com diferentes multiplicadores de Lagrange foi calculado, considerando o erro obtido e o BD-Rate de cada faixa de redução de tempo analisada. Assim, à medida que o valor de λ aumenta, a qualidade do vídeo passa a impactar de maneira mais expressiva no valor do custo.

$$J = Erro + \lambda.BD_{(Rate)} \tag{17}$$

Os resultados, obtidos através das sequências *station2_1080p25_60f*, *pan_hdr_amazon_1080p* e *seaplane_hdr_amazon_1080p24_60f*, com os quatro CQs recomendados em (DAEDE; NORKIN; BRAILOVSKIY, 2020), são mostrados na Tabela 50. Para valores maiores de λ, o modo “número ocorrência ponderada” apresentou melhor desempenho para quatro faixas de redução de complexidade analisadas: RTC alvo de 20%, RTC alvo de 40%, RTC alvo de 50% e RTC alvo de

60%. Além disso, este modo apresentou segundo menor custo para o RTC alvo de 30% e RTC alvo de 70%, ficando atrás apenas do modo "número ocorrência" e SI-TI, respectivamente.

É importante destacar que para o uso do modo SI-TI, o algoritmo que calcula as informações espaciais e temporais teria que ser incorporado ao algoritmo de controle, gerando um acréscimo no tempo de processamento. No caso do RTC alvo de 70%, estima-se que o cálculo das informações espaciais e temporais de uma sequência HD 1080, com 180 quadros, equivale a um *overhead* de 0,35%. Este valor é, aproximadamente, 23 vezes maior que o *overhead* médio inserido pelo algoritmo do controlador *baseline*. Já para a resolução UHD 4K o *overhead* inserido pelo cálculo SI-TI é de 0,42%, ou seja, cerca de 85 vezes maior que o *overhead* médio inserido pelo algoritmo do controlador *baseline*.

Assim, diante do que foi discutido até aqui, a especialização do controlador de acordo com características do vídeo de entrada foi realizada com base em aprendizado de máquina, implementando o modelo no controlador através do modo "número de ocorrência ponderada".

Tabela 50 – Modos de classificação.

RTC_10%									
Modo	**Erro ±(p.p.)**	**BD-Rate (%)**	**Custo**						
			λ = 1	**λ = 0,8**	**λ = 0,6**	**λ = 0,4**	**λ = 0,2**	**λ = 0,05**	**λ = 0,01**
nº ocorrência	3,65	3,26	6,91	6,26	5,61	4,95	4,30	3,81	3,68
nº ocor. pond.	4,78	5,22	10,00	8,95	7,91	6,87	5,82	5,04	4,83
média	3,13	2,94	6,07	5,48	4,89	4,30	3,72	3,28	3,16
SI-TI	2,59	2,50	5,09	4,59	4,09	3,59	3,09	2,72	2,62
RTC_20%									
Modo	**Erro ±(p.p.)**	**BD-Rate (%)**	**Custo**						
			λ = 1	**λ = 0,8**	**λ = 0,6**	**λ = 0,4**	**λ = 0,2**	**λ = 0,05**	**λ = 0,01**
nº ocorrência	1,24	6,83	8,07	6,71	5,34	3,98	2,61	1,59	1,31
nº ocor. pond.	1,94	4,20	6,14	5,30	4,46	3,62	2,78	2,15	1,98
média	1,94	9,07	11,01	9,19	7,38	5,57	3,75	2,39	2,03
SI-TI	1,85	4,96	6,81	5,82	4,83	3,84	2,84	2,10	1,90
RTC_30%									
Modo	**Erro ±(p.p.)**	**BD-Rate (%)**	**Custo**						
			λ = 1	**λ = 0,8**	**λ = 0,6**	**λ = 0,4**	**λ = 0,2**	**λ = 0,05**	**λ = 0,01**
nº ocorrência	0,89	8,18	9,08	7,44	5,80	4,17	2,53	1,30	0,97
nº ocor. pond.	1,36	8,50	9,86	8,16	6,46	4,76	3,06	1,79	1,45
média	1,74	10,12	11,86	9,83	7,81	5,79	3,76	2,24	1,84
SI-TI	2,78	6,98	9,76	8,37	6,97	5,57	4,18	3,13	2,85
RTC_40%									
Modo	**Erro ±(p.p.)**	**BD-Rate (%)**	**Custo**						
			λ = 1	**λ = 0,8**	**λ = 0,6**	**λ = 0,4**	**λ = 0,2**	**λ = 0,05**	**λ = 0,01**
nº ocorrência	1,37	16,15	17,51	14,29	11,06	7,83	4,60	2,18	1,53
nº ocor. pond.	3,67	11,02	14,69	12,48	10,28	8,08	5,87	4,22	3,78
média	2,20	15,06	17,26	14,24	11,23	8,22	5,21	2,95	2,35
SI-TI	1,00	16,29	17,30	14,04	10,78	7,52	4,26	1,82	1,17
RTC_50%									
Modo	**Erro ±(p.p.)**	**BD-Rate (%)**	**Custo**						
			λ = 1	**λ = 0,8**	**λ = 0,6**	**λ = 0,4**	**λ = 0,2**	**λ = 0,05**	**λ = 0,01**
nº ocorrência	1,88	21,49	23,37	19,07	14,77	10,47	6,17	2,95	2,09
nº ocor. pond.	3,62	17,50	21,12	17,62	14,12	10,62	7,12	4,49	3,79
média	3,54	20,00	23,55	19,55	15,55	11,55	7,54	4,54	3,74
SI-TI	3,38	18,91	22,29	18,51	14,73	10,94	7,16	4,32	3,57

Continuação da Tabela 50

RTC_60%

Modo	Erro ±(p.p.)	BD-Rate (%)	Custo						
			λ = 1	λ = 0,8	λ = 0,6	λ = 0,4	λ = 0,2	λ = 0,05	λ = 0,01
nº ocorrência	2,46	35,55	38,01	30,90	23,79	16,68	9,57	4,24	2,82
nº ocor. pond.	3,01	28,42	31,43	25,74	20,06	14,38	8,69	4,43	3,30
média	6,18	36,86	43,05	35,67	28,30	20,93	13,56	8,03	6,55
SI-TI	2,07	31,19	33,26	27,02	20,79	14,55	8,31	3,63	2,38

RTC_70%

Modo	Erro ±(p.p.)	BD-Rate (%)	Custo						
			λ = 1	λ = 0,8	λ = 0,6	λ = 0,4	λ = 0,2	λ = 0,05	λ = 0,01
nº ocorrência	2,72	53,71	56,43	45,69	34,94	24,20	13,46	5,40	3,26
nº ocor. pond.	3,51	49,91	53,42	43,44	33,46	23,48	13,50	6,01	4,01
média	5,30	53,90	59,19	48,41	37,63	26,85	16,08	7,99	5,84
SI-TI	3,28	48,45	51,73	42,04	32,35	22,66	12,97	5,70	3,76

APÊNDICE K – Materiais sobre o teste subjetivo

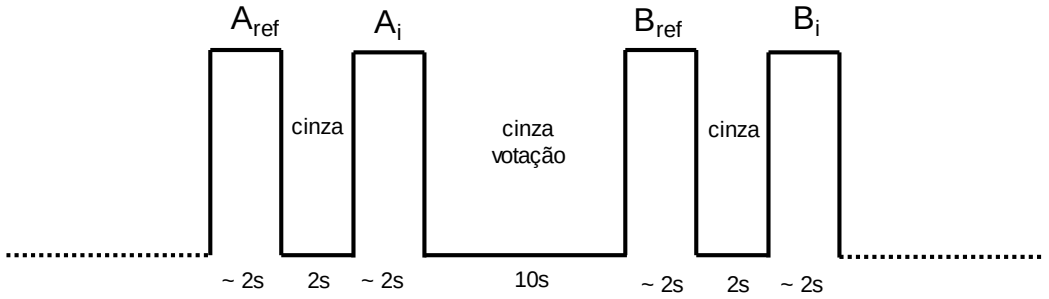
Olá!

Você participará de testes subjetivos de qualidade de vídeo digital. Primeiramente, será aplicado um teste de acuidade visual e um teste de daltonismo. Caso os resultados sejam satisfatórios, após a explicação detalhada do procedimento de teste, será realizado um treinamento para certificar sua compreensão em relação ao processo. Por fim, os testes serão efetuados. Estima-se que a duração de todas as etapas citadas seja de, aproximadamente, 12 minutos.

Por favor, preencha os seguintes dados:

Nome:
Profissão:
Idade:
Usa óculos: ( ) sim ( ) não

Para efetuar os testes, você verá cinco vídeos diferentes. Cada vídeo terá quatro experimentos, os quais devem ser avaliados conforme exemplifica a Figura abaixo.



Assim, para cada experimento, o seguinte processo é realizado:

1) O vídeo 1 (referência - A_{ref}) é apresentado;
2) A tela fica cinza por 2 segundos;
3) O vídeo 2 (A_i) é apresentado;
4) A tela fica cinza durante 10 segundos e você deve realizar a votação.

Ao final de cada processo você deve avaliar o comprometimento na qualidade do vídeo 2 em relação ao vídeo 1. Você expressará seu julgamento considerando a seguinte escala:

( ) Imperceptível

( ) Pouco perceptível

( ) Moderadamente perceptível

( ) Perceptível

( ) Muito perceptível

Vamos treinar como você deve proceder!

Vídeo em análise: ducks_take_off_1080p50_60f

Avaliação:

( ) Imperceptível

( ) Pouco perceptível

( ) Moderadamente perceptível

( ) Perceptível

( ) Muito perceptível

VAMOS COMEÇAR!

Vídeo em análise: "*sequência de teste*"

Avaliação_Vídeo 1:

( ) Imperceptível
( ) Pouco perceptível
( ) Moderadamente perceptível
( ) Perceptível
( ) Muito perceptível

Avaliação_ Vídeo 2:
( ) Imperceptível
( ) Pouco perceptível
( ) Moderadamente perceptível
( ) Perceptível
( ) Muito perceptível

Avaliação_ Vídeo 3:
( ) Imperceptível
( ) Pouco perceptível
( ) Moderadamente perceptível
( ) Perceptível
( ) Muito perceptível

Avaliação_ Vídeo 4:
( ) Imperceptível
( ) Pouco perceptível
( ) Moderadamente perceptível
( ) Perceptível
( ) Muito perceptível

** As sequências de teste estão disponíveis em: https://drive.google.com/drive/folders/0AIcCfpEnCHBmUk9PVA.

APÊNDICE L – Resultados de cada sequência de vídeo analisada

Tabela 51 – *Netflix_Boat*

RTC alvo (%)	**Versão Baseline**				**CCAM**			
	RTC real (%)	**Erro (± p.p.)**	**BD-Rate (%)**	**Overhead 10^{-6} (%)**	**RTC real (%)**	**Erro (± p.p.)**	**BD-Rate (%)**	**Overhead 10^{-6} (%)**
10	12,35	2,35	0,09	91,23	10,39	0,39	0,56	109,83
20	22,37	2,37	2,68	96,19	20,22	0,22	2,47	118,85
30	35,75	5,75	8,60	124,89	30,17	0,17	5,70	132,22
40	47,61	7,61	10,45	152,17	40,37	0,37	8,25	142,75
50	58,32	8,32	8,08	167,91	50,53	0,53	8,71	175,92
60	66,57	6,57	9,92	207,24	60,82	0,82	11,23	219,19
70	75,71	5,71	15,96	274,10	68,68	1,32	12,51	259,76
Média	**-**	**15,61 (%)**	**7,97**	**159,10**	**-**	**5,46 (%)**	**7,06**	**165,50**

Tabela 52 – *Netflix_FoodMarket*

RTC alvo (%)	**Versão Baseline**				**CCAM**			
	RTC real (%)	**Erro (± p.p.)**	**BD-Rate (%)**	**Overhead 10^{-6} (%)**	**RTC real (%)**	**Erro (± p.p.)**	**BD-Rate (%)**	**Overhead 10^{-6} (%)**
10	13,04	3,04	0,11	71,52	11,52	1,52	4,17	82,90
20	24,04	4,04	4,35	73,38	19,92	0,08	10,22	86,84
30	35,82	5,82	10,83	87,72	29,19	0,81	21,02	81,10
40	46,92	6,92	13,16	119,44	37,47	2,53	25,44	110,55
50	53,81	3,81	20,26	121,86	47,76	2,24	41,96	121,47
60	65,33	5,33	30,73	163,16	57,38	2,62	64,18	153,15
70	79,22	9,22	71,64	233,14	66,68	3,32	73,81	178,44
Média	**-**	**16,71 (%)**	**21,58**	**124,32**	**-**	**5,47 (%)**	**34,40**	**116,35**

Tabela 53 – *park_joy*

RTC alvo (%)	Versão Baseline				CCAM			
	RTC real (%)	**Erro (± p.p.)**	**BD-Rate (%)**	**Overhead 10^{-6} (%)**	**RTC real (%)**	**Erro (± p.p.)**	**BD-Rate (%)**	**Overhead 10^{-6} (%)**
10	12,54	2,54	0,40	56,23	13,54	3,54	0,74	65,87
20	23,25	3,25	1,61	58,54	23,86	3,86	2,58	73,62
30	32,88	2,88	3,27	71,62	32,34	2,34	3,77	80,86
40	48,18	8,18	5,08	92,96	43,21	3,21	4,53	87,11
50	56,59	6,59	7,35	102,36	52,40	2,40	6,88	96,75
60	64,20	4,20	9,17	111,39	61,04	1,04	9,06	129,26
70	70,56	0,56	12,36	143,97	68,54	1,46	11,69	133,20
Média	**-**	**13,24 (%)**	**5,61**	**91,01**	**-**	**11,31 (%)**	**5,28**	**95,24**

Tabela 54 – *Netflix_PierSeaside*

RTC alvo (%)	Versão Baseline				CCAM			
	RTC real (%)	Erro (± p.p.)	BD-Rate (%)	Overhead 10^{-6} (%)	RTC real (%)	Erro (± p.p.)	BD-Rate (%)	Overhead 10^{-6} (%)
10	14,80	4,80	1,35	128,64	14,99	4,99	4,69	151,24
20	22,67	2,67	1,95	129,78	21,74	1,74	6,02	156,26
30	32,73	2,73	4,28	148,26	30,18	0,18	10,49	146,93
40	43,67	3,67	7,99	199,89	37,90	2,10	18,77	197,99
50	57,15	7,15	12,89	241,14	50,16	0,16	23,29	209,32
60	68,98	8,98	24,56	304,16	59,73	0,27	28,95	270,09
70	77,12	7,12	48,05	380,31	68,10	1,90	36,93	318,96
Média	**-**	**17,00 (%)**	**14,44**	**218,88**	**-**	**9,70 (%)**	**18,45**	**207,26**

Tabela 55 – *rush_hour*

RTC alvo (%)	Versão Baseline				CCAM			
	RTC real (%)	Erro (± p.p.)	BD-Rate (%)	Overhead 10^{-6} (%)	RTC real (%)	Erro (± p.p.)	BD-Rate (%)	Overhead 10^{-6} (%)
10	10,96	0,96	0,30	89,86	8,96	1,04	0,26	100,49
20	22,70	2,70	2,04	91,05	18,26	1,74	1,42	108,84
30	34,75	4,75	4,28	106,53	28,66	1,34	3,10	99,09
40	47,52	7,52	6,84	147,65	38,61	1,39	7,74	138,65
50	50,35	0,35	13,33	153,24	48,09	1,91	31,01	155,37
60	60,78	0,78	17,29	180,13	58,07	1,93	36,97	178,73
70	78,21	8,21	57,72	279,29	68,98	1,02	56,58	214,81
Média	**-**	**10,21 (%)**	**14,54**	**149,68**	**-**	**14,83 (%)**	**19,58**	**142,28**

Tabela 56 – *speed_bag*

RTC alvo (%)	Versão Baseline				CCAM			
	RTC real (%)	**Erro (± p.p.)**	**BD-Rate (%)**	**Overhead 10^{-6} (%)**	**RTC real (%)**	**Erro (± p.p.)**	**BD-Rate (%)**	**Overhead 10^{-6} (%)**
10	15,46	5,46	3,34	62,55	8,94	1,06	20,87	82,40
20	27,48	7,48	6,61	74,12	18,99	1,01	30,05	89,62
30	36,60	6,60	19,06	83,94	29,17	0,83	43,54	102,85
40	49,61	9,61	38,67	103,27	39,27	0,73	64,87	120,97
50	62,65	12,65	129,72	135,53	49,78	0,22	82,49	139,42
60	65,07	5,07	168,24	144,13	59,06	0,94	121,56	171,33
70	64,56	5,44	168,24	140,63	65,65	4,39	159,05	204,20
Média	**-**	**25,65 (%)**	**60,94**	**106,31**	**-**	**4,07 (%)**	**74,63**	**130,11**

Tabela 57 – *Netflix_Dancer*

RTC alvo (%)	Versão Baseline				CCAM			
	RTC real (%)	Erro ($\pm$ p.p.)	BD-Rate (%)	Overhead 10^{-6} (%)	RTC real (%)	Erro ($\pm$ p.p.)	BD-Rate (%)	Overhead 10^{-6} (%)
10	12,86	2,86	0,01	32,34	10,91	0,91	0,56	29,74
20	22,99	2,99	0,04	34,67	20,97	0,97	3,91	32,56
30	34,79	4,79	1,05	40,65	28,47	1,53	15,54	34,63
40	47,25	7,25	2,79	53,22	38,39	1,61	21,32	43,47
50	55,49	5,49	7,60	53,63	49,31	0,69	33,87	50,01
60	64,34	4,34	29,64	59,64	59,23	0,77	34,51	55,98
70	76,01	6,01	43,16	82,94	68,41	1,59	55,04	67,47
Média	**-**	**14,92 (%)**	**12,03**	**51,01**	**-**	**4,00 (%)**	**14,61**	**44,84**

Tabela 58 – *Netflix_WindAndNature*

RTC alvo (%)	**Versão Baseline**				**CCAM**			
	RTC real (%)	**Erro (± p.p.)**	**BD-Rate (%)**	**Overhead 10^{-6} (%)**	**RTC real (%)**	**Erro (± p.p.)**	**BD-Rate (%)**	**Overhead 10^{-6} (%)**
10	12,42	2,42	1,12	47,97	10,03	0,03	0,56	41,31
20	22,38	2,38	2,12	53,10	20,92	0,92	4,53	40,29
30	34,37	4,37	6,79	62,59	32,31	2,31	8,36	55,51
40	46,69	6,69	11,09	73,78	41,85	1,85	19,61	66,46
50	55,93	5,93	18,11	85,98	49,28	0,72	35,47	76,18
60	66,38	6,38	26,64	101,67	59,93	0,07	57,53	85,50
70	76,99	6,99	70,17	150,12	69,92	0,08	74,52	102,70
Média	**-**	**14,27 (%)**	**19,43**	**82,17**	**-**	**2,70 (%)**	**28,65**	**66,85**

Tabela 59 – *sol_levante_aom_sdr_519-649*

RTC alvo (%)	**Versão Baseline**				**CCAM**			
	RTC real (%)	**Erro (± p.p.)**	**BD-Rate (%)**	**Overhead 10^{-6} (%)**	**RTC real (%)**	**Erro (± p.p.)**	**BD-Rate (%)**	**Overhead 10^{-6} (%)**
10	20,85	10,85	0,45	33,78	20,38	10,38	12,25	33,21
20	26,53	6,53	1,40	40,19	27,35	7,35	25,84	35,05
30	35,49	5,49	4,81	45,09	31,82	1,82	23,28	31,84
40	44,62	4,62	10,46	47,67	42,22	2,22	33,87	44,53
50	57,68	7,68	24,54	60,92	53,20	3,20	41,15	54,59
60	70,85	10,85	84,78	86,01	62,36	2,36	74,38	71,80
70	75,14	5,14	136,47	91,49	71,02	1,02	99,34	85,60
Média	**-**	**30,25 (%)**	**37,56**	**57,88**	**-**	**23,43 (%)**	**44,30**	**50,95**

Tabela 60 – *sparks_aom_sdr_6026-6502*

RTC alvo (%)	**Versão Baseline**				**CCAM**			
	RTC real (%)	**Erro (± p.p.)**	**BD-Rate (%)**	**Overhead 10^{-6} (%)**	**RTC real (%)**	**Erro (± p.p.)**	**BD-Rate (%)**	**Overhead 10^{-6} (%)**
10	15,46	5,46	0,39	22,42	12,81	2,81	0,34	22,69
20	23,99	3,99	0,79	31,02	23,14	3,14	0,55	24,06
30	36,65	6,65	1,54	32,45	32,15	2,15	2,04	24,08
40	46,82	6,82	2,69	36,47	41,96	1,96	5,46	30,84
50	55,69	5,69	8,15	45,89	51,07	1,07	21,8	37,65
60	69,40	9,40	14,82	64,27	61,22	1,22	44,67	47,46
70	77,11	7,11	24,2	87,22	72,25	2,25	39,11	61,11
Média	**-**	**21,56 (%)**	**7,51**	**45,68**	**-**	**9,04 (%)**	**16,28**	**35,41**

Tabela 61 – *FlowerSky_A*

RTC alvo (%)	Versão Baseline				CCAM			
	RTC real (%)	Erro ($\pm$ p.p.)	BD-Rate (%)	Overhead 10^{-6} (%)	RTC real (%)	Erro ($\pm$ p.p.)	BD-Rate (%)	Overhead 10^{-6} (%)
10	13,15	3,15	0,07	4,13	12,49	2,49	0,58	4,75
20	21,77	1,77	0,16	4,53	20,43	0,43	0,94	5,24
30	31,76	1,76	0,37	5,37	30,56	0,56	1,29	6,14
40	44,55	4,55	0,79	6,41	41,05	1,05	2,36	9,06
50	53,62	3,62	1,96	7,38	50,85	0,85	3,57	9,59
60	63,95	3,95	4,47	8,75	61,35	1,35	5,65	11,63
70	73,48	3,48	6,63	11,50	67,71	2,29	4,82	11,37
Média	**-**	**10,92 (%)**	**2,06**	**6,87**	**-**	**5,54 (%)**	**2,74**	**8,26**

Anexos

ANEXO A – Tabela Snellen

Teste de Snellen www.provisu.ch