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

Dissertação

Aprendizado de Máquina Aplicado à Redução do Custo Computacional do Test Zone Search e da Estimação de Movimento Affine do Codificador VVC

Ramiro Gomes da Silva Viana

Pelotas, 2025

Ramiro Gomes da Silva Viana

Aprendizado de Máquina Aplicado à Redução do Custo Computacional do Test Zone Search e da Estimação de Movimento Affine do Codificador VVC

Dissertação 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 Mestre em Ciência da Computação.

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

Pelotas, 2025

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação da Publicação

V614a Viana, Ramiro Gomes da Silva

Aprendizado de máquina aplicado à redução do custo computacional do Test Zone Search e da estimação de movimento Affine do codificador VVC [recurso eletrônico] / Ramiro Gomes da Silva Viana ; Luciano Volcan Agostini, orientador ; Marcelo Schiavon Porto, Guilherme Ribeiro Corrêa, coorientadores. — Pelotas, 2025.
98 f. : il.

Dissertação (Mestrado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2025.

1. Predição inter-quadros. 2. Versatile video coding. 3. Aprendizado de máquina. 4. Árvores de decisão. I. Agostini, Luciano Volcan, orient. II. Porto, Marcelo Schiavon, coorient. III. Corrêa, Guilherme Ribeiro, coorient. IV. Título.

CDD 005

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

Ramiro Gomes da Silva Viana

Aprendizado de Máquina Aplicado à Redução do Custo Computacional do Test Zone Search e da Estimação de Movimento Affine do Codificador VVC

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

Data da Defesa: 11 de Março de 2025

Banca Examinadora:

Prof. Dr. Luciano Volcan Agostini (orientador)
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

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

Prof. Dr. Fabio Luis Livi Ramos
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Gustavo Freitas Sanchez
Doutor em Computação pela Pontifícia Universidade Católica do Rio Grande do Sul.

Dedico este trabalho à minha mãe Mara.

AGRADECIMENTOS

Primeiramente, agradeço à minha mãe, Mara Silva, por ser uma pessoa maravilhosa, por todo amor que me deu, pela sua dedicação em me educar e por me proporcionar a oportunidade e o incentivo para estudar. O apoio dela foi fundamental para toda a minha vida, especialmente a minha formação, incluindo o desenvolvimento deste trabalho. Não teria chegado até aqui sem ela, e parte desta conquista também é dela.

Aos meus avós Teresinha Silva e Francisco Silva (*in memoriam*), por estarem presentes, sempre que possível, em todos os momentos de minha vida.

Ao professor Luciano Agostini, meu orientador, pelos valiosos conhecimentos que compartilhou comigo e por todas as oportunidades que me proporcionou desde o início da minha trajetória na Iniciação Científica, quando eu ainda estava na graduação em 2021. Agradeço também pela confiança depositada em mim ao me orientar no mestrado e por ter visto meu potencial como pesquisador científico.

Aos professores Guilherme Corrêa e Marcelo Porto, meus co-orientadores, que foram fundamentais na minha jornada científica e acadêmica ao longo destes dois anos de mestrado.

Aos amigos e colegas que conheci durante o mestrado e no ViTech, com quem convivi ao longo desse período. Agradecimentos especiais a Cristiano Santos, Denis Maass, Eduardo Costa, Gilberto Kriesler, Gustavo Rehbein, Iago Storch, Patrick Rosa, Marcello Muñoz, Ruhan Conceição, Victor Costa e Vítor Costa pela parceria e companheirismo dentro e fora do laboratório, pelo trabalho conjunto em projetos e artigos, além das viagens para eventos científicos que fizemos juntos. Também agradeço a Marta Loose, Fernando Sagrilo e Rafael Ferreira, com quem trabalhei desde o meu período de Iniciação Científica e que contribuíram significativamente para o desenvolvimento de projetos e artigos científicos.

Por fim, quero agradecer também ao grupo ViTech, o qual eu tenho prazer de participar desde que iniciei a Iniciação Científica.

People think of education as something they can finish.
— ISAAC ASIMOV

RESUMO

VIANA, Ramiro Gomes da Silva. **Aprendizado de Máquina Aplicado à Redução do Custo Computacional do Test Zone Search e da Estimação de Movimento Affine do Codificador VVC**. Orientador: Luciano Volcan Agostini. 2025. 98 f. Dissertação (Mestrado em Ciência da Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2025.

À medida que a demanda por transmissão de vídeo aumenta com trabalho remoto, educação e serviços de *streaming*, a necessidade de avanços contínuos nas tecnologias de codificação de vídeo torna-se cada vez mais evidente. Adaptar-se às exigências crescentes de entrega e consumo eficiente de vídeos requer o desenvolvimento e aprimoramento constante nos padrões de codificação de vídeo, com o *Versatile Video Coding* (VVC) emergindo como um exemplo notável. Este trabalho apresenta uma visão geral dos principais algoritmos na Predição Inter-Quadros do VVC, com foco principalmente no *Test Zone Search* (TZS) e na Estimação de Movimento *Affine* (AME), duas das ferramentas mais intensivas em termos de computação dentro do VVC. Além disso, este trabalho introduz uma abordagem para acelerar o TZS e a AME usando Aprendizado de Máquina, especificamente utilizando Árvores de Decisão. Primeiro, foi realizada uma aceleração no *software* de referência *VVC Test Model* (VTM) focada no TZS, pulando seletivamente as suas três últimas etapas, utilizando um conjunto de 12 Árvores de Decisão, uma para cada tamanho de bloco suportado pelo TZS no VVC. Em seguida, foi realizada a aceleração da AME do VTM, considerando a implementação acelerada do TZS. Neste caso, todo o processo da AME é pulado seletivamente, utilizando um novo conjunto de 12 Árvores de Decisão, uma para cada tamanho de bloco suportado pela AME no VVC. Esta abordagem proposta alcançou uma redução média de 20,99% no tempo total de codificação do VVC, uma redução média de 62,15% no tempo de execução do TZS e uma redução média de 63,58% no tempo de execução da AME, ocasionando em uma pequena perda média de eficiência de BD-BR de somente 0,90%. Estes resultados são competitivos quando comparados com os trabalhos da literatura e demonstram que a estratégia de uso de aprendizado de máquina para reduzir o custo computacional do VVC tem potencial de seguir gerando resultados expressivos em soluções futuras.

Palavras-chave: predição inter-quadros; versatile video coding; aprendizado de máquina; árvores de decisão.

ABSTRACT

VIANA, Ramiro Gomes da Silva. **Machine Learning Applied to Reducing the Computational Cost of Test Zone Search and Affine Motion Estimation in the VVC Encoder**. Advisor: Luciano Volcan Agostini. 2025. 98 f. Dissertation (Masters in Computer Science) – Technology Development Center, Federal University of Pelotas, Pelotas, 2025.

As the demand for video transmission surges on remote work, education, and streaming services, the need for continuous advancements in video encoding technologies becomes increasingly evident. Adapting to the evolving requirements of efficient video delivery and consumption necessitates ongoing development and enhancement in video encoding standards, with Versatile Video Coding (VVC) emerging as a notable example. This work provides an overview of the main algorithms in VVC Inter-Frame Prediction, focusing primarily on Test Zone Search (TZS) and Affine Motion Estimation (AME), two of the most computationally intensive tools within VVC. Furthermore, this work introduces an approach to accelerate TZS and AME using Machine Learning, specifically employing Decision Trees. First, an acceleration of the *VVC Test Model* (VTM) reference software was performed, focusing on TZS, by selectively skipping its last three steps using a set of 12 Decision Trees, one for each block size supported by TZS in VVC. Next, the acceleration of the VTM's AME was performed, considering the accelerated implementation of TZS. In this case, the entire AME process is selectively skipped using a new set of 12 Decision Trees, one for each block size supported by AME in VVC. The proposed approach achieved an average reduction of 20.99% in the total VVC encoding time, an average reduction of 62.15% in the TZS execution time, and an average reduction of 63.58% in the AME execution time, resulting in a small average BD-BR efficiency loss of only 0.90%. These results are competitive compared to related works in the literature and demonstrate that the strategy of using machine learning to reduce the computational cost of VVC has the potential to continue yielding significant results in future solutions.

Keywords: inter frame prediction; versatile video coding; machine learning; decision trees.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AME	Estimação de Movimento *Affine*, do inglês *Affine Motion Estimation*
AMVP	Predição Avançada do Vetor de Movimento, do inglês *Advanced Motion Vector Prediction*
AMVR	Resolução de Vetor de Movimento Adaptável, do inglês *Adaptive Motion Vector Resolution*
BCW	Bi-predição com Pesos a nível de Unidade de Codificação do inglês *Bi-prediction with CU-level Weights*
BDOF	Fluxo Óptico Bidirecional, do inglês *Bi-directional Optical Flow*
BD-BR	*Bjøntegaard Delta-Bitrate*
BD-PSNR	*Bjøntegaard Delta-PSNR*
CB	Bloco de Codificação, do inglês *Coding Block*
CIIP	Predição Inter/Intra-Quadro Combinada, do inglês *Combined Inter/Intra-picture Prediction*
CNN	Rede Neural Convolucional, do inglês *Convolutional Neural Network*
CP	Ponto de Controle, do inglês *Control Point*
CU	Unidade de Codificação, do inglês *Coding Unit*
CSV	Valores Separados Por Virgula, do inglês *Comma-Separated Values*
CTB	Bloco de Árvore de Codificação, do inglês *Coding Tree Block*
CTCs	Condições Comuns de Teste, do inglês *Common Test Conditions*
CTU	Unidade de Árvore de Codificação, do inglês *Coding Tree Unit*
DMVR	Refinamento de Vetor de Movimento no Decodificador, do inglês *Decoder-side MV Refinement*
DT	Árvore de Decisão, do inglês *Decision Tree*
FHD	Alta Definição Completa, do inglês *Full High Definition*
FME	Estimação de Movimento Fracionária, do inglês *Fractional Motion Estimation*
FS	Busca Completa, do inglês *Full Search*
G	Gradiente

GOP	Grupo de Quadros, do inglês *Group of Pictures*
GPM	Modo de Particionamento Geométrico, do inglês *Geometric Partitioning Mode*
H	Horizontal
HD	Alta Definição, do inglês *High Definition*
HEVC	*High Efficiency Video Coding*
IEEE	*Institute of Electrical and Electronics Engineers*
IME	Estimação de Movimento Inteira, do inglês *Integer Motion Estimation* item[JVET] *Joint Video Experts Team*
L0	Lista 0
L1	Lista 1
MC	Compensação de Movimento, do inglês *Motion Compensation*
ME	Estimação de Movimento, do inglês *Motion Estimation*
MV	Vetor de Movimento, do inglês *Motion Vector*
MVP	Predição do Vetor de Movimento, do inglês *Motion Vector Prediction*
MVD	Diferença do Vetor de Movimento, do inglês *Motion Vector Difference*
ML	Aprendizado de Máquina, do inglês *Machine Learning*
MPEG	*Moving Picture Experts Group*
PDI	Processamento Digital de Imagens
PSNR	*Peak Signal-to-Noise Ratio*
QHD	Alta Definição Quádrupla, do inglês *Quad High Definition*
QP	Parâmetro de Quantização, do inglês *Quantization Parameter*
QTBT	Árvore Binária com Árvore de Tipos Múltiplos Aninhada, do inglês *Quad Tree Binary Tree*
QTMT	Árvore Quádrupla com Árvore de Tipos Múltiplos Aninhada, do inglês *Quad-tree with nested Multi-type Tree*
RA	*Random Access*
RF	Floresta Aleatória, do inglês, *Random Forest*
RGB	Vermelho, Verde e Azul, do inglês, *Red, Green and Blue*
RSL	Revisão Sistemática da Literatura
SA	Área de Busca, do inglês, *Search Area*
SAD	Soma das Diferenças Absolutas, do inglês, *Sum of Absolute Differences*
SCLIP	Particionamento Inter baseado em Aprendizado Contrastivo Supervisionado, do inglês *Supervised-Contrastive-Learning-based Inter Partitioning*
SD	Definição Padrão, do inglês, *Standard Definition*

SMVD	Codificação MVD Simétrica, do inglês, *Symmetric MVD Coding*
SPIE	*Society of Photo-Optical Instrumentation Engineers*
SR	Intervalo de Busca, do inglês, *Search Range*
SVH	Sistema Visual Humano
TR	Redução de Tempo, do inglês *Time Reduction*
TZS	Pesquisa na Zona de Teste, do inglês *Test Zone Search*
UHD	Ultra Alta Definição, do inglês *Ultra High Definition*
UVG	*Ultra Video Group*
V	Vertical
VVenC	*Fraunhofer Versatile Video Encoder*
VVC	*Versitile Video Coding*
VTM	*VVC Test Model*
WQVGA	*Wide Quarter Video Graphics Array*
WVGA	*Wide Video Graphics Array*
YCbCr	Luminância, Crominância Azul e Crominância Vermelha, do inglês, *Luminance, Chrominance Blue and Chrominance Red*
μ	Média
σ^2	Variância

SUMÁRIO

1 INTRODUÇÃO

Atualmente, observa-se um aumento significativo na demanda por tecnologias que suportem o processamento de vídeos digitais. Este aumento é especialmente notável dada a contagem global de usuários de *internet*, que atingiu mais de 5 bilhões de pessoas, com aproximadamente 90% do tráfego da *internet* sendo atribuído a vídeos digitais (Cisco, 2023) devido às diversas plataformas de *streaming* disponíveis para entretenimento, reuniões virtuais e trabalho remoto. E esta demanda aumentou ainda mais por causa da pandemia do COVID-19 (Mercat et al., 2021). Por causa da grande complexidade na transmissão e armazenamento desses vídeos desde o seu surgimento, o desenvolvimento de codificadores tornou-se, desde então, imperativo. Eles são projetados para comprimir dados de forma eficiente enquanto mantêm a qualidade de imagem. Isso, por sua vez, simplifica a manipulação e transmissão de vídeos digitais. Por outro lado, para alcançar uma compressão eficiente, a codificação de vídeo requer um enorme esforço computacional. Tudo isso levou a mais pesquisas sobre o processo de codificação de vídeo (Palau et al., 2021).

Nos últimos anos, o cenário da tecnologia de vídeo tem experimentado um crescimento substancial, com a popularização das resoluções 4k *Ultra High Definition* (UHD) e 8k *Full Ultra High Definition* (FHD) (Park et al., 2022). Consequentemente, vários grupos têm dedicado esforços para desenvolver novos codificadores de vídeo que possam manter a eficiência em seus processos de codificação. Uma adição notável a esse conjunto de codificadores é o *Versatile Video Coding* (VVC), que foi oficialmente lançado em julho de 2020 e é o atual padrão de codificação de vídeo do estado da arte da *ISO* e *ITU-T* (Chen et al., 2023).

O VVC apresenta uma substancial melhoria na eficiência de codificação, com uma redução notável na taxa de bits de até 50% para uma qualidade visual equivalente em comparação com seu antecessor, o *High Efficiency Video Coding* (HEVC) (Bross et al., 2021). Para tanto, o VVC introduz uma série de ferramentas de codificação inovadoras em sua arquitetura. Estas incluem estruturas de particionamento flexíveis, um novo conjunto de transformadas, estimação de movimentos não translacionais e várias outras. No entanto, essa melhoria é acompanhada por um aumento muito alto no custo computacional, resultando em um tempo de codificação mais lento para o VVC em comparação com o HEVC (Siqueira; Correa; Grellert, 2020).

A Predição Inter-Quadros do VVC incorpora ferramentas essenciais para otimizar a codificação de vídeo, como o *Test Zone Search* (TZS) e a Estimação de Movimento *Affine*, do inglês *Affine Motion Estimation* (AME), que exploram a redundância espacial para otimizar a codificação de vídeo. O TZS é conhecido por sua eficiência no processo de Estimação de Movimento, do inglês *Motion Estimation* (ME), e é o algoritmo de casamento de blocos mais comumente usado nos codificadores atuais (Ferreira et al., 2023). No entanto, o TZS continua sendo um dos algoritmos mais demorados dentro do VVC (Ferreira et al., 2022), em especial, pelo fato de possuir quatro etapas de execução. A AME é uma nova ferramenta de codificação introduzida na predição Inter-Quadros do VVC, sendo crucial para melhorar a eficiência ao estimar movimentos complexos, como o redimensionamento, a rotação e o cisalhamento (Maass et al., 2023). No entanto, a AME aumenta significativamente o esforço computacional do VVC, resultando em um tempo total de codificação mais longo no processo de predição (Loose et al., 2022).

Uma das abordagens atuais com maior potencial para reduzir o custo computacional dos codificadores de vídeo é o Aprendizado de Máquina, do inglês *Machine Learning* (ML). Diversos trabalhos na literatura apresentam soluções para diferentes codificadores e suas respectivas ferramentas, utilizando técnicas variadas de Aprendizado de Máquina, como a Árvore de Decisão, do inglês *Decision Tree* (DT). A Árvore de Decisão se destaca pela sua simplicidade, facilidade de interpretação e implementação direta, tornando-se uma ferramenta crucial na ciência de dados (Somvanshi et al., 2016).

1.1 Objetivos

O objetivo principal deste trabalho é apresentar propostas de aceleração nas ferramentas *Test Zone Search* e na Estimação de Movimento *Affine* presentes na Predição Inter-Quadros do VVC utilizando Aprendizado de Máquina (mais especificamente Árvores de Decisão) para reduzir seu custo computacional com perdas tão pequenas quanto possíveis na eficiência de codificação.

1.2 Organização do Trabalho

Este trabalho está organizado da seguinte forma: No Capítulo 2 são apresentados os conceitos básicos relacionados à compressão de vídeo digital. No Capítulo 3 são abordados os princípios básicos do codificador de vídeo VVC e as ferramentas principais presentes na sua Predição Inter-Quadros. No Capítulo 4 são descritos os fundamentos básicos relacionados ao Aprendizado de Máquina, com foco nas Árvores de Decisão. No Capítulo 5 é apresentada a Revisão Sistemática da Literatura realizada para este trabalho. No Capítulo 6 é apresentada uma avaliação estatística das ferramentas presentes na Predição Inter-Quadros do VVC. No Capítulo 7 é apresentada a metodologia estabelecida para este trabalho, assim como os modelos de Aprendizado de Máquina desenvolvidos para reduzir o custo computacional da Predição Inter-Quadros do VVC. No Capítulo 8 são apresentados os resultados obtidos com os modelos de Aprendizado de Máquina desenvolvidos. Por fim, no Capítulo 9 são apresentadas as conclusões obtidas neste trabalho desenvolvido.

2 CONCEITOS DE COMPRESSÃO DE VÍDEO DIGITAL

Neste capítulo são apresentados os conceitos básicos relacionados a vídeos digitais e as etapas necessárias para realizar a compressão destes vídeos, com foco na Predição Inter-Quadros.

Os princípios da compressão de vídeo digital, também chamada de codificação de vídeo, se originaram no Processamento Digital de Imagens (PDI). O Processamento Digital de Imagens tem sido um assunto de pesquisa crucial há mais de cinco décadas, coincidindo com o advento de computadores capazes de armazenar imagens. A aplicação das técnicas de PDI pode ser amplamente categorizada em duas áreas principais: aprimorar informações visuais para interpretação humana e processar dados de imagem para armazenamento, transmissão e representação, levando em consideração a percepção automática por máquinas (Gonzalez; Woods, 2010). PDI pode ser delineado por três objetivos primários: codificação, restauração-aperfeiçoamento e análise de imagem (Coady et al., 2019). Dentro do objetivo de codificação, o computador manipula a imagem em uma forma que elimina a redundância de dados, visando minimizar os requisitos de armazenamento e comunicação.

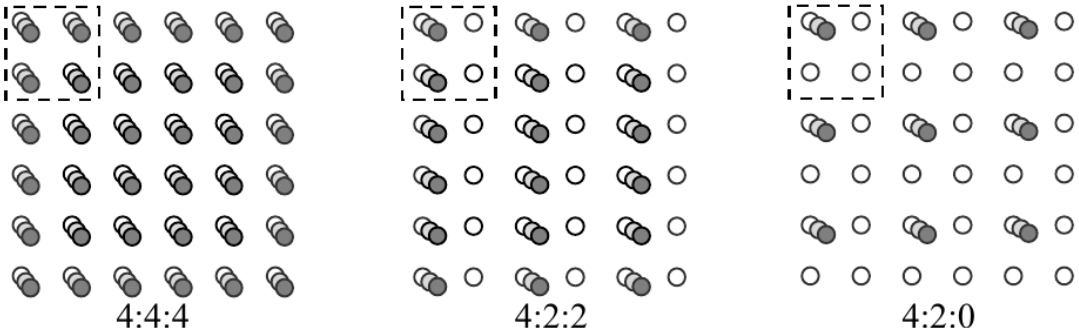
2.1 Conceitos Básicos de Vídeo Digital

Um vídeo digital pode ser entendido como uma sequência de imagens estáticas que, quando exibidas em sequência, geram a sensação de movimento das imagens sendo projetadas no vídeo. Estas imagens estáticas presentes nos vídeos são chamadas de quadros. Os quadros são divididos em blocos, e estes blocos são divididos em píxeis (Corrêa, 2014). O pixel, abreviação de *Picture Element*, é uma unidade básica de uma imagem ou vídeo digital composta por três canais de informação (Richardson, 2002). Estes canais podem ser representados por diversos espaços de cores, como, por exemplo, RGB representando as cores vermelho (R), verde (G) e azul (B), ou YCbCr representando luminância (Y), crominância azul (Cb) e crominância vermelha (Cr) (Bhaskaran; Konstantinides, 1997). A luminância representa a intensidade luminosa ou o brilho no pixel, enquanto as crominâncias azul e vermelha são os com-

ponentes das suas respectivas cores (Porto, 2012).

O sistema visual humano é mais sensível a informações de luminância do que crominância (Agostini, 2007). Para explorar essa característica, os codificadores de vídeo tipicamente usam o espaço de cores do tipo YCbCr. Nesse espaço de cores, é possível aplicar uma operação de subamostragem nas amostras de crominância, que são menos importantes para o sistema visual humano. Nesse caso, existem diferentes formatos de subamostragem, sendo os mais comuns o 4:4:4, 4:2:2 e 4:2:0 (Porto, 2012). No formato 4:4:4, para cada quatro amostras de luminância, há quatro amostras de crominância azul e quatro amostras de crominância vermelha, ou seja, não há subamostragem. No formato 4:2:2, para cada quatro amostras de luminância, existem duas amostras de crominância azul e duas amostras de crominância vermelha. Por fim, no formato 4:2:0, para cada quatro amostras de luminância, há apenas uma amostra de crominância azul e uma amostra de crominância vermelha (Corrêa, 2014). Este último é o formato utilizado neste trabalho, por ser o formato mais comum nos dispositivos atuais (Singh; Ahamed, 2018). A Figura 1 apresenta esses três formatos de subamostragem.

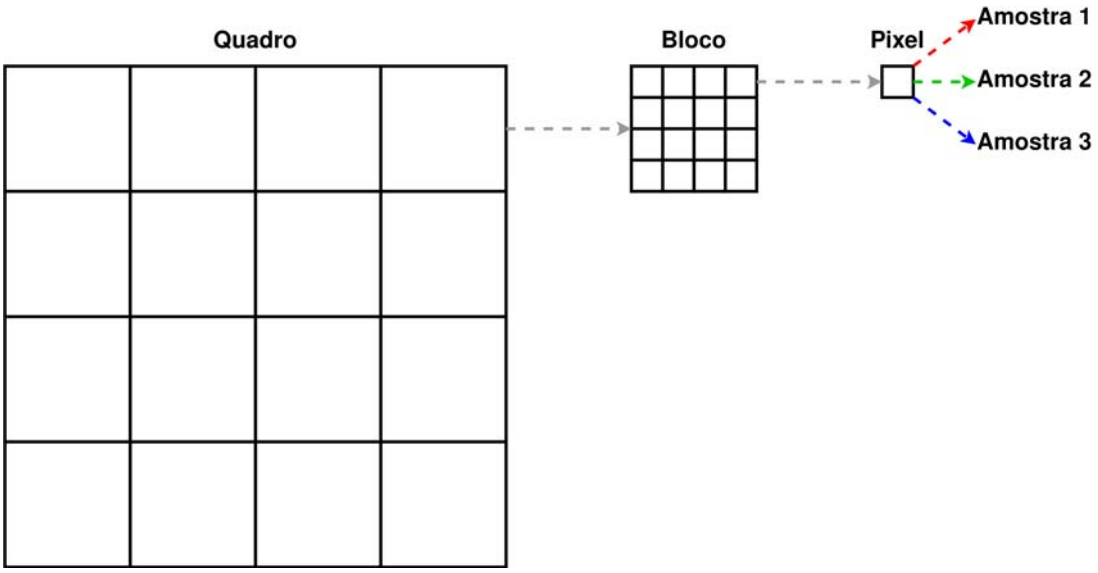
Figura 1 – Formatos de subamostragem de crominância.



Fonte: Adaptada de Richardson (2002).

Cada informação de cor em um pixel é chamada de amostra. Um exemplo genérico do particionamento de um quadro em um bloco e depois em um pixel para os três canais está ilustrado na Figura 2.

Figura 2 – Exemplo de particionamento de um quadro de um vídeo digital.



Fonte: Adaptada de Gonçalves (2021).

O processamento digital de imagens é realizado a partir da avaliação de características das imagens que serão usadas na definição das operações que se seguirão. Neste contexto, a extração das características tipicamente considera diversas relações entre os valores dos píxeis (ou das amostras) e as relações mais usadas são: Média (μ), Variância (σ^2) e Gradiente (G) (Lindino et al., 2022).

A Média (μ) mede a concentração dos dados em uma distribuição. Pode ser calculada pela Equação (1), onde W é a largura do bloco de imagem, H é a altura do bloco de imagem e $X(i,j)$ são os valores dos píxeis do bloco de imagem atual.

$$\mu = \frac{1}{W \cdot H} \sum_{i=1}^{W} \sum_{j=1}^{H} X(i,j) \tag{1}$$

A Variância (σ^2) mede a dispersão que identifica a distância de cada valor em relação ao valor médio. Ao calcular a variância de um bloco, é possível identificar a homogeneidade de seus píxeis. Pode ser calculada pela Equação (2).

$$\sigma^2 = \frac{1}{W \cdot H} \sum_{i=1}^{W} \sum_{j=1}^{H} (X(i,j) - \mu)^2 \tag{2}$$

O Gradiente (G) identifica a mudança na intensidade dos píxeis em uma imagem ao longo de uma direção específica e pode ser medido horizontalmente e verticalmente. Utilizando o Filtro Sobel (Sobel, 2014), é possível calcular diferenças finitas, fornecendo uma aproximação do gradiente da intensidade dos píxeis na imagem. Através das Equações 3 e 4, os valores dos Gradientes Horizontal (G_x) e Vertical (G_y) podem ser obtidos para um ponto de pixel dado no bloco, com $M(i,j)$ sendo a matriz original

desse ponto. Para obter os valores totais dos Gradientes no bloco, os valores absolutos de todos os Gradientes calculados por essas equações são somados, como mostrado nas Equações 5 e 6.

$$G_x(i,j) = M_{i,j} \cdot \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{3}$$

$$G_y(i,j) = M_{i,j} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{4}$$

$$G_x = \sum_{i=1}^{W} \sum_{j=1}^{H} abs(G_x(i,j)) \tag{5}$$

$$G_y = \sum_{i=1}^{W} \sum_{j=1}^{H} abs(G_y(i,j)) \tag{6}$$

Estas características também podem ser utilizadas em vídeos digitais, tendo em vista que um vídeo é uma sucessão de imagens.

A quantidade de píxeis presente nos quadros, tanto na posição horizontal (H) quanto na vertical (V), define a resolução dos vídeos digitais. As resoluções mais utilizadas na atualidade são mostradas na Tabela 1.

Tabela 1 – Resoluções mais comuns dos vídeos digitais.

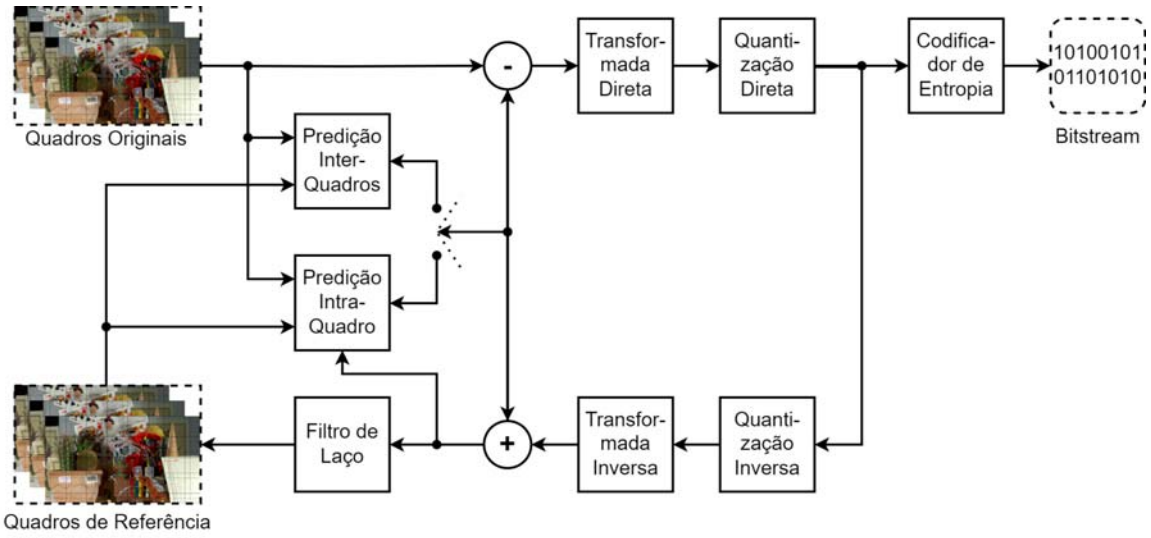
Qualidade	**Sigla**	**Resolução**	**Quantidade de píxeis (H $\times$ V)**
8k Full Ultra High Definition	8k FHD	4320p	7680 píxeis $\times$ 4320 píxeis
4k Ultra High Definition	4k UHD	2160p	3840 píxeis $\times$ 2160 píxeis
2k Quad High Definition	2k QHD	1440p	2560 píxeis $\times$ 1440 píxeis
Full High Definition	FHD	1080p	1920 píxeis $\times$ 1080 píxeis
High Definition	HD	720p	1280 píxeis $\times$ 720 píxeis
Wide Video Graphics Array	WVGA	480p	830 píxeis $\times$ 480 píxeis
Wide Quarter Video Graphics Array	WQVGA	480p	376 píxeis $\times$ 480 píxeis
Standard Definition	SD	360p	640 píxeis $\times$ 360 píxeis

A resolução do vídeo digital não comprimido impacta diretamente no tamanho do arquivo que o representa, com resoluções maiores gerando arquivos muito grandes, devido a uma quantidade maior de dados presente em cada quadro. Por causa disso, a codificação de vídeo se torna cada vez mais essencial, enquanto o seu processo se torna mais complexo.

2.2 Codificador de Vídeo Híbrido

Tendo em vista que um vídeo é composto por um número muito grande de imagens estáticas, codificar um vídeo exige muito tempo, esforço computacional e consumo de energia devido às diversas etapas necessárias para serem executadas por cada codificador. Essas etapas incluem Predição Inter-Quadros, Predição Intra-Quadro, Transformadas Direta e Inversa, Quantização Direta e Inversa, Filtro de Laço e Codificador de Entropia (Agostini, 2007), como demonstrado na Figura 3, sendo que este tipo de codificador de vídeo é chamado de codificador híbrido, por explorar diversos tipos de redundância (Bhaskaran; Konstantinides, 1997).

Figura 3 – Diagrama de blocos da estrutura básica do codificador de vídeo genérico.



Fonte: Adaptada de Agostini (2007).

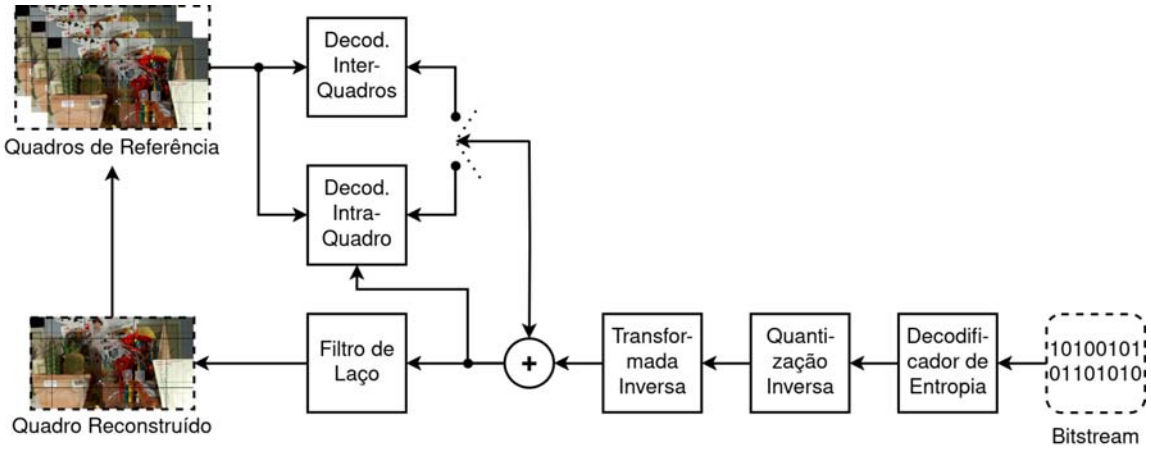
- **Predição Inter-Quadros:** Explora a redundância temporal do quadro atualmente sendo codificado em relação a quadros de referência previamente codificados, reusando dados disponíveis nestes quadros de referência para representar o quatro atual.
- **Predição Intra-Quadro:** Explora a redundância espacial presente no quadro atualmente sendo codificado, reusando dados de blocos que já foram codificados para representar o bloco atual.
- **Transformada:** Transforma os dados dos resíduos resultantes das predições do domínio espacial para o domínio das frequências. Os resíduos são as diferenças entre as predições e o quadro original.

- **Quantização:** Atenua ou elimina as frequências menos relevantes para o Sistema Visual Humano (SVH) de forma adaptativa através de um parâmetro externo. Com isso é possível definir a taxa de compressão e as perdas de qualidade em um codificador de vídeos. Para o VVC, este parâmetro é chamado de Parâmetro de Quantização, do inglês *Quantization Parameter* (QP).
- **Codificador de Entropia:** Comprime os valores resultantes da quantização em um processo sem perdas, com o objetivo de representar os símbolos mais usados na saída da quantização com a menor quantidade de bits possível, reduzindo a redundância entrópica.
- **Filtro de Laço:** Serve para melhorar a qualidade subjetiva dos vídeos, sendo necessária pois o processo de codificação insere artefatos nas imagens, tais como: efeito de bloco, do inglês *blocking*, granulação, do inglês *ringing*, e desfoque, do inglês *blurring*.

Os quadros de referência são gerados através da etapa de reconstrução, que está presente no processo de decodificação para garantir que o codificador e o decodificador usem exatamente as mesmas referências. Esta etapa é composta pela Quantização Inversa e Transformada Inversa (Porto, 2012).

O sinal de *bitstream* gerado pelo codificador é o sinal a ser transmitido para fazer as transmissões dos vídeos digitais. Depois de enviado ao seu destino, ele deve ser decodificado para ser reconstruído como um vídeo digital. Para realizar a decodificação, também é necessário executar diferentes etapas, incluindo Decodificador de Entropia, Quantização Inversa, Transformada Inversa, Decodificação Inter-Quadros, Decodificação Intra-Quadro e Filtro de Laço (Agostini, 2007), como mostrado na Figura 4, sendo muito parecido com a etapa de reconstrução do quadro de codificação.

Figura 4 – Diagrama de blocos da estrutura básica do decodificador de vídeo genérico.



Fonte: Adaptada de Agostini (2007).

Apesar de ser um processo mais simples do que o codificador, o decodificador deve estar apto a interpretar qualquer decisão tomada pelo codificador e, assim, deve implementar todas as ferramentas previstas pelo padrão. Por outro lado, o codificador pode ser implementado com alguma simplificação para fins de redução de custo computacional, por exemplo, desde que gere um *bitstream* que seja possível de ser decodificado (Palau et al., 2021).

2.3 Predição Inter-Quadros

A Predição Inter-Quadros, foco deste trabalho, é uma das etapas mais importantes presentes nos codificadores de vídeo por ser a responsável direta pela redução da redundância temporal (Porto, 2012), que é extremamente presente em vídeos digitais (Chien et al., 2021). A redundância temporal ocorre quando quadros temporalmente vizinhos em um vídeo possuem dados muito semelhantes, como ilustrado na Figura 5 usando dois quadros da sequência de vídeo *Cactus_1920x1080_50*.

Figura 5 – Exemplo de redundância temporal em quadros vizinhos de um vídeo.

(a) Quadro Atual.

(b) Quadro Seguinte.

Fonte: Elaborada pelo autor.

Com os quadros vizinhos sendo muito similares, é possível codificar os blocos do quadro atual reutilizando dados de blocos de quadros previamente codificados, chamados quadros de referência, preservando a informação visual, mas reduzindo a quantidade de dados necessários para representar essa informação (Palau et al., 2021). Para isso, os codificadores de vídeo utilizam algoritmos de busca para encontrar, dentro de uma região específica de blocos nos quadros de referência, chamada área de busca, do inglês *Search Area* (SA), as melhores correspondências entre o bloco do quadro atual e os blocos dos quadros de referência, reduzindo assim a quantidade de dados necessária para representar os blocos codificados (Viana et al., 2024a). O tamanho da área de busca é definido de acordo com o intervalo de busca, do inglês *Search Range* (SR), que representa os seus limites numéricos, ou seja, o deslocamento máximo permitido para a procura de um bloco correspondente.

Nos codificadores atuais, existem algoritmos específicos para a busca de blocos. Um dos mais utilizados é a Busca Completa, do inglês *Full Search* (FS), presente em diversos codificadores de vídeo (Doan et al., 2017). Esse algoritmo testa todas as opções de blocos na área de busca e, assim, possui um custo computacional muito elevado. Além do FS, há também diversos algoritmos rápidos de busca, que visam reduzir o custo computacional do processo através da aplicação de uma ou mais heurísticas para acelerar a convergência da busca. Atualmente um dos algoritmos mais importantes nesta categoria é a Pesquisa na Zona de Teste, do inglês *Test Zone Search* (TZS), que é foco neste trabalho. O TZS alcança resultados muito próximos aos da Busca Completa, porém com um custo computacional significativamente menor (Gonçalves et al., 2017).

Os algoritmos de busca comparam o bloco atual com os blocos candidatos na área de busca aplicando algum critério de similaridade. O mais usado é a Soma das Diferenças Absolutas, do inglês *Sum of Absolute Differences* (SAD), que é calculada através da Equação (7), em que Cur representa o bloco atual e Ref representa o bloco de referência, ambos com tamanho de bloco de $M \times N$ amostras(Gonçalves et al., 2022).

$$SAD = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} |Cur(i,j) - Ref(i,j)| \tag{7}$$

A redução da redundância temporal é alcançada através de dois estágios principais: a Estimação de Movimento, do inglês *Motion Estimation* (ME), e a Compensação de Movimento, do inglês *Motion Compensation* (MC) (Saldanha et al., 2020).

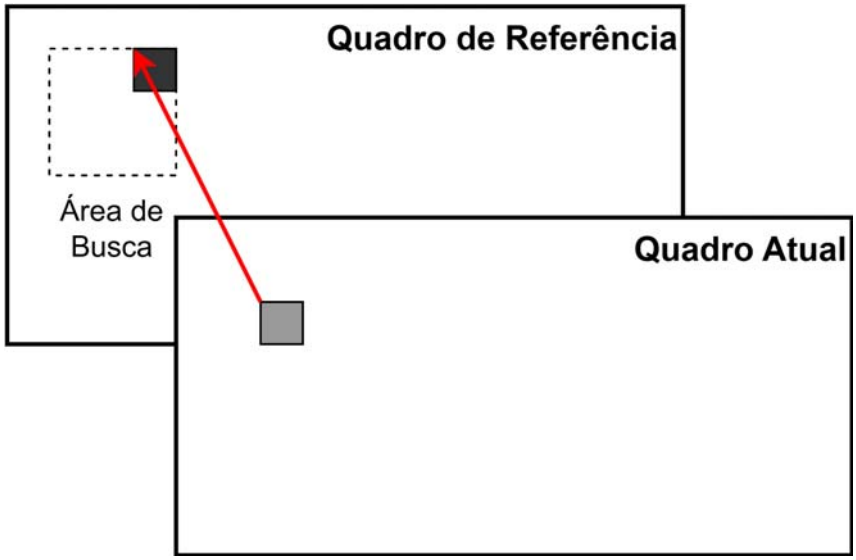
- **Estimação de Movimento:** Para cada Bloco de Codificação, do inglês *Coding Block* (CB), de luminância do quadro de entrada, busca o bloco com melhor correspondência, ou casamento, em quadros de referência. Quando este bloco é encontrado, pelo menos um Vetor de Movimento, do inglês *Motion Vector* (MV), é gerado para indicar a localização do bloco no quadro de referência dentro do intervalo de busca (Singh; Ahamed, 2018).

- **Compensação de Movimento:** Opera para complementar à ME, copiando os blocos de melhor correspondência dos quadros de referência e monta o quadro predito a partir dos Vetores de Movimento gerados pela ME. Essa etapa é necessária porque a ME nem sempre encontra um casamento perfeito entre blocos e, nestes casos, o erro de predição não deve ser descartado. Assim a MC gera o bloco predito, que é subtraído do bloco atual, gerando o que é chamado de resíduo, que é processado pelas demais etapas do codificador.

Os codificadores modernos possuem ferramentas aprimoradas para a ME. Um exemplo introduzido nos codificadores de vídeo atuais é a Estimação de Movimento *Affine*, do inglês *Affine Motion Estimation* (AME), capaz de estimar diversos tipos de movimento, como zoom, rotação, ajuste de proporção de tela e cisalhamento (Maass et al., 2023). A AME, assim como o TZS, é uma das principais ferramentas abordadas neste trabalho.

A Figura 6 mostra um diagrama de exemplo genérico da redução da redundância temporal entre quadros vizinhos. Neste exemplo, o bloco cinza claro representa o bloco a ser codificado no quadro atual, enquanto o bloco cinza escuro representa o bloco predito após o processamento do algoritmo de busca.

Figura 6 – Exemplo de aplicação da redundância temporal em quadros vizinhos de um vídeo.



Fonte: Adaptada de Storch (2020).

A Predição Inter-Quadros é aplicada em todos os codificadores de vídeo disponíveis no mercado atualmente, incluindo o *Versatile Video Coding* (VVC) (Bross et al., 2020) que é um dos codificadores mais importantes atualmente e foco deste trabalho.

3 *VERSATILE VIDEO CODING*

Neste capítulo são apresentados os conceitos básicos do codificador de vídeo VVC e as ferramentas principais presentes na sua Predição Inter-Quadros, o *Test Zone Search* e a Estimação de Movimento *Affine*.

O atual padrão de codificador de vídeo estado da arte é o *Versatile Video Coding* (VVC) (Hamidouche et al., 2022), desenvolvido pelo ISO/IEC *Moving Picture Experts Group* (MPEG) e disponibilizado em julho de 2020 (Chen et al., 2023). O VVC alcança uma economia média de 44,4% na taxa de bits quando comparado ao seu predecessor, o padrão *High Efficiency Video Coding* (HEVC) (Siqueira; Correa; Grellert, 2020), para uma mesma qualidade objetiva da imagem. Apesar dos requisitos computacionais significativos, o VVC oferece taxas de compressão notáveis e supera outros codificadores disponíveis comercialmente em termos de eficiência de codificação (Bross et al., 2021). Para isso, são usadas ferramentas e algoritmos avançados, resultando em uma compressão de vídeo altamente eficiente (Loose et al., 2022).

Para desenvolver o VVC e avaliar possíveis modificações no padrão, foi desenvolvido seu software de referência chamado de *VVC Test Model* (VTM) (JVET - Joint Video Experts Team, 2019). O VTM é disponibilizado gratuitamente e foi empregado no desenvolvimento deste trabalho.

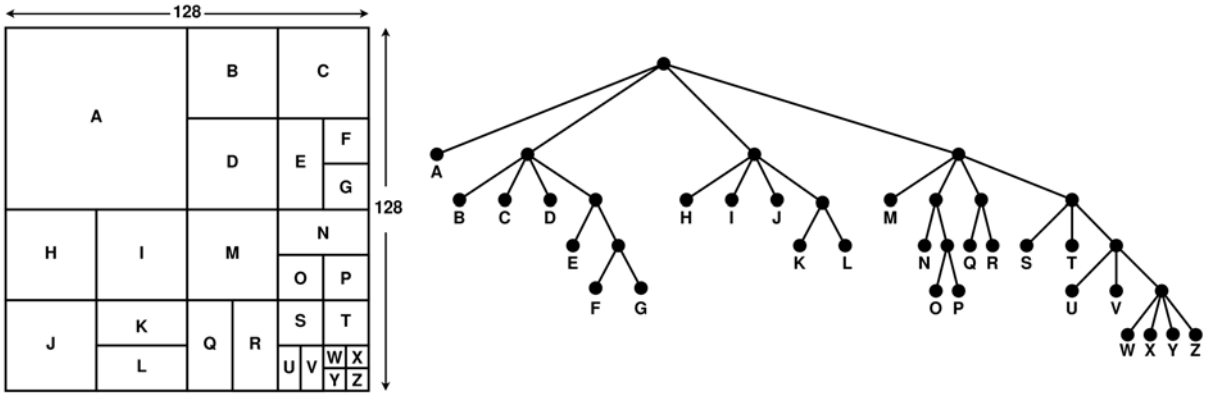
Quando o VVC codifica um vídeo, os quadros a serem codificados são primeiramente particionados em blocos denominados Unidades de Árvore de Codificação, do inglês *Coding Tree Units* (CTU), contendo um número de píxeis variando de 32×32 até 128×128. Cada CTU é composta por três Blocos de Árvore de Codificação, do inglês *Coding Tree Block* (CTB), um para cada componente de cor, ou seja, um para luminância, um para crominância azul e um para crominância vermelha (Sun et al., 2019).

Atuando como a raiz de uma estrutura *Quadtree*, cada CTU é subdividida em Unidades de Codificação, do inglês *Coding Units* (CU). As CUs possuem até 28 tamanhos diferentes, variando de 4×4 píxeis até 128×128 píxeis, sendo também utilizadas para melhor adequar o processo de codificação às características da imagem (Saldanha et al., 2020). De forma similar à CTU, cada CU é composta por três Blocos de Codifi-

cação, do inglês *Coding Blocks* (CB), que representam cada um dos componentes de cor YCbCr (Sun et al., 2019). As CBs são os blocos que realmente são processados pelas ferramentas de codificação. No escopo deste trabalho, CB e bloco serão usados como sinônimos.

A Figura 7 mostra um exemplo genérico de uma CTU sendo particionada diversas vezes, em tamanhos de bloco diferentes.

Figura 7 – Exemplo de particionamento de blocos em um quadro de uma sequência de vídeo.



Fonte: Elaborada pelo autor.

3.1 Predição Inter-Quadros do VVC

No VVC, a Predição Inter-Quadros consome quase metade do tempo total de codificação (Siqueira; Correa; Grellert, 2021). Nela, a ME é realizada em dois estágios principais: a Estimação de Movimento Inteira, do inglês *Integer Motion Estimation* (IME), e a Estimação de Movimento Fracionária, do inglês *Fractional Motion Estimation* (FME).

- **Estimação de Movimento Inteira:** É a primeira a ser executada e é aplicada às amostras inteiras dos quadros de referência na busca do melhor casamento. Um aspecto fundamental para a eficiência da IME são os algoritmos de busca, com destaque para o TZS (Doan et al., 2017).
- **Estimação de Movimento Fracionária:** É utilizada como a segunda etapa da ME e, uma vez encontrada a melhor correspondência inteira, é feita uma interpolação para gerar amostras fracionárias e sobre essas amostras fracionárias é feito uma busca para um novo melhor casamento.

Existem dois modos principais para a ME no VVC: Predição Unidirecional e Predição Bidirecional, sendo que, em ambos os modos, 27 tamanhos de bloco são utilizados, variando de 4×8 até 128×128 (Chen; Ye; Kim, 2020).

- **Predição Unidirecional:** É o modo básico de ME no VVC, onde a predição utiliza apenas blocos de um quadro de referência como a melhor correspondência.
- **Predição Bidirecional:** É o modo capaz de referenciar dois blocos em diferentes quadros de referência, ao realizar uma predição que avalia a combinação entre dois blocos para gerar a predição final (Bross et al., 2021).

Assim como em outros padrões de codificadores de vídeo, o VVC permite a codificação de quadros fora de ordem. Dessa forma, os quadros de referência utilizados na Predição Inter-Quadros podem ser quadros anteriores ou posteriores na ordem de apresentação, desde que todos tenham sido previamente codificados (Bross et al., 2021). Os quadros temporalmente anteriores e posteriores ao quadro que está sendo codificado são armazenados em dois *buffers*, chamados de Lista 0 (L0) e Lista 1 (L1).

Além do TZS e da AME, VVC possui também outras ferramentas importantes que são utilizadas para executar a Predição Inter-Quadros, incluindo:

- Modos *Merge* Estendidos, do inglês *Extended Merge Modes (Bross et al., 2021)*;
- Modo de Particionamento Geométrico, do inglês *Geometric Partitioning Mode* (GPM) (Gao et al., 2021);
- Predição Inter/Intra-Quadro Combinada, do inglês *Combined Inter/Intra-picture Prediction* (CIIP) (Browne; Ye; Kim, 2022);
- Predição Avançada do Vetor de Movimento, do inglês *Advanced Motion Vector Prediction* (AMVP) (Li et al., 2021);
- Resolução de Vetor de Movimento Adaptável, do inglês *Adaptive Motion Vector Resolution* (AMVR) (Liu et al., 2019);
- Bi-predição com Pesos a nível de Unidade de Codificação, do inglês *Bi-prediction with CU-level Weights* (BCW) (Bross et al., 2021);
- Fluxo Óptico Bidirecional, do inglês *Bi-Directional Optical Flow* (BDOF) (Bross et al., 2021);
- Refinamento do Vetor de Movimento no Decodificador, do inglês *Decoder-side MV Refinement* (DMVR) (Bross et al., 2021).

Estas ferramentas não são foco deste trabalho e, por isso, não serão detalhadas neste texto.

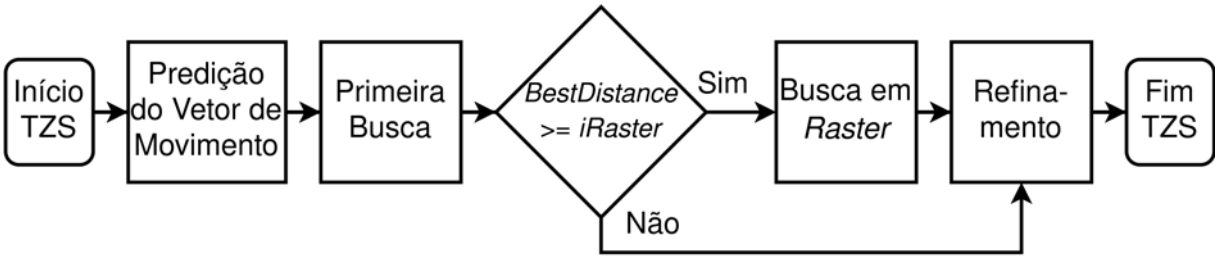
Nas próximas seções, serão discutidos em mais detalhes o *Test Zone Search* e a Estimação de Movimento *Affine*, que são focos deste trabalho.

3.2 *Test Zone Search*

O *Test Zone Search* é um algoritmo rápido de busca que visa encontrar o melhor casamento de blocos em quadros de referência, explorando a redundância temporal em áreas vizinhas de um bloco candidato (Martins et al., 2017), sendo uma das partes mais demoradas do processo de codificação no VVC. Ele é um algoritmo incremental de busca da ME e foi projetado para executar uma busca adaptativa do Vetor de Movimento. O TZS é ideal para evitar a ocorrência de mínimos locais no processo de busca (Gonçalves et al., 2022) e utiliza apenas 12 tamanhos diferentes de CBs: 16×16, 16×32, 16×64, 32×16, 32×32, 32×64, 64×16, 64×32, 64×64, 64×128, 128×64 e 128×128 (Viana et al., 2024a). Assim como em outros algoritmos de busca, no TZS a busca é realizada na matriz de amostras de luminância e, assim que o bloco com melhor casamento é encontrado no quadro de referência, sua posição é usada tanto para gerar o vetor de movimento da CB de luminância, quanto para gerar os vetores de movimento das duas CBs de crominância associadas. Nesse caso, se existe subamostragem de cor, então o vetor de movimento da CB de luminância é escalado proporcionalmente para gerar o vetor para as CBs de crominância.

Ele foi introduzido no padrão *High Efficiency Video Coding* (HEVC) e foi herdado pelo seu sucessor, o VVC (Sant'anna et al., 2021). O TZS executa quatro etapas: Predição do Vetor de Movimento, Primeira Busca, Busca em *Raster* e Refinamento (Ferreira et al., 2022), conforme apresentado na Figura 8. Cada uma destas etapas será discutida nas próximas subseções.

Figura 8 – Fluxograma das etapas do TZS.

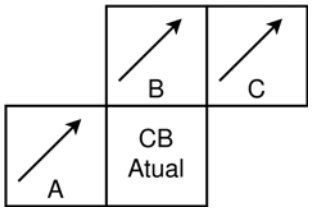


Fonte: Adaptada de Viana et al. (2024a).

3.2.1 Predição do Vetor de Movimento

A Predição do Vetor de Movimento, do inglês *Motion Vector Prediction* (MVP), é uma etapa obrigatória na codificação do VVC (Viana et al., 2025). Ela faz uma predição do vetor de movimento inicial do processo de busca escolhendo entre cinco preditores possíveis: o esquerdo (*A*), o superior (*B*), o superior direito (*C*), o preditor zero e o preditor mediano. A Figura 9 apresenta os preditores A, B e C, em relação à CB atual.

Figura 9 – Vetores de movimento adjacentes à unidade de previsão atual.



Fonte: Elaborada pelo autor.

O preditor mediana é calculado através da Equação (8) (Gonçalves et al., 2022).

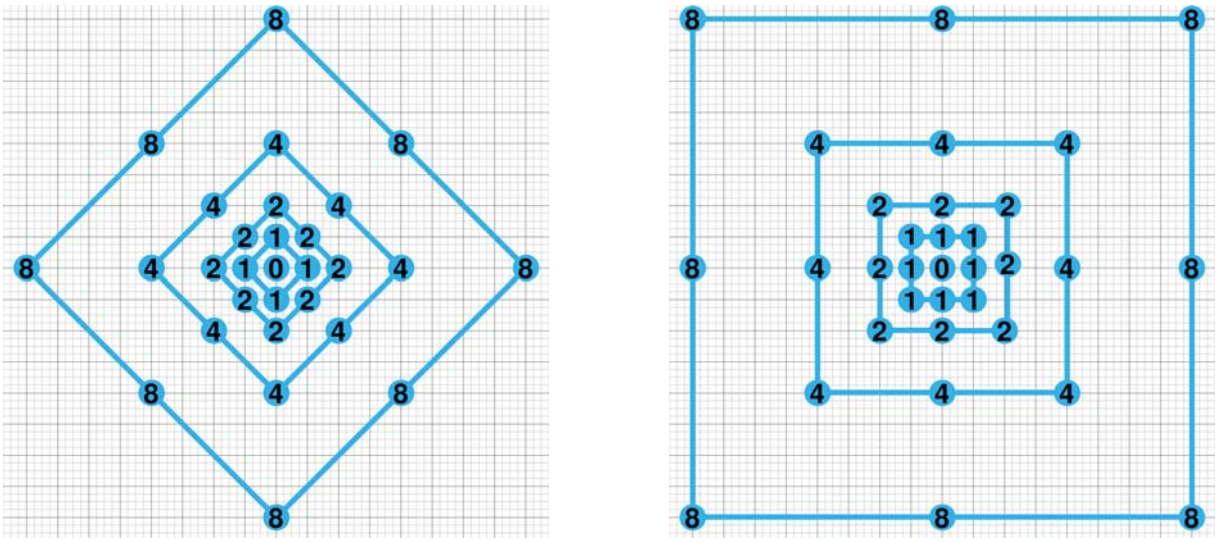
$$Mediana = A + B + C - Min(A, Min(B, C)) - Max(A, Max(B, C)) \quad (8)$$

A MVP define o melhor preditor ao escolher aquele que alcança os melhores resultados de acordo com um critério de similaridade. Normalmente, o critério SAD é utilizado.

3.2.2 Primeira Busca

A Primeira Busca, do inglês *First Search*, é uma busca expandida aplicada à área de busca. Ele inicia o processo de busca em oito pontos, empregando os padrões de busca em diamante ou quadrado, como mostrados na Figura 10.

Figura 10 – Padrões da Primeira Busca.



(a) Busca em Diamante. (b) Busca em Quadrado.

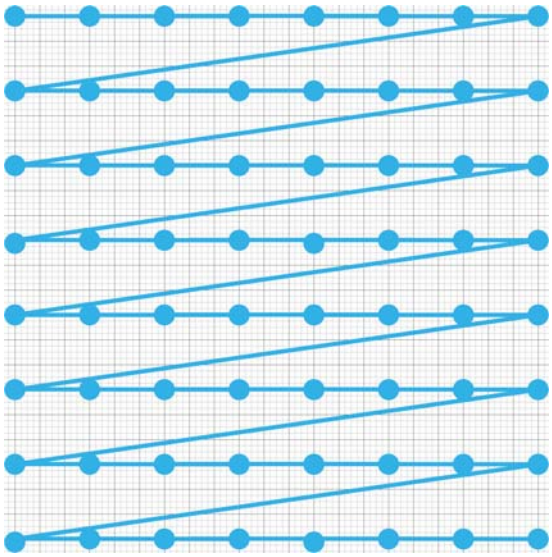
Fonte: Adaptada de Viana et al. (2024a).

O ponto central da busca (0, 0) é determinado pela MVP, com a área de busca sendo expandida usando potências de dois (Vayalil; Paul; Kong, 2017), com o número de expansões permitidas sendo restrito a seis (Ferreira et al., 2023). Durante este processo, o SAD do bloco central é comparado com o SAD dos novos blocos candidatos. O bloco com o menor SAD entre todos os blocos avaliados é designado como a escolha ótima nesta fase, com os blocos candidatos devendo estar dentro da área de busca definida (Gonçalves et al., 2017).

3.2.3 Busca em *Raster*

Em seguida, é realizada a Busca em *Raster*, do inglês *Raster Search*, que é a etapa de maior custo computacional do TZS, uma vez que faz uma busca completa com subamostragem dentro de um intervalo de busca (SR), limitado, conforme ilustrado na Figura 11.

Figura 11 – Padrão da Busca em *Raster*.



Fonte: Adaptada de Viana et al. (2024a).

A Busca em *Raster* só é implementada se a distância do melhor bloco encontrado (*BestDistance*), identificada na segunda etapa, exceder a taxa de subamostragem *raster* (*iRaster*); caso contrário, a Busca em *Raster* será pulada (Ferreira et al., 2023). O *iRaster* pode variar de 5 a 8 e é aplicado tanto horizontalmente quanto verticalmente (Sant'anna et al., 2021). O total de n blocos candidatos comparados na Busca em *Raster* é definido na Equação (9).

$$n = \left[\frac{1 + (SR \cdot 2)}{iRaster}\right]^2 \tag{9}$$

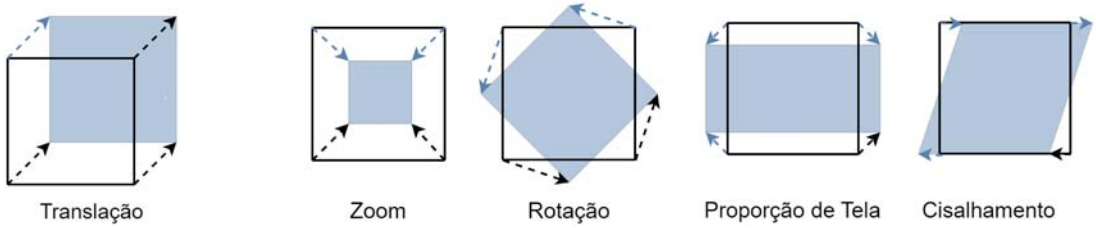
3.2.4 Refinamento

A última etapa do TZS é o Refinamento, que visa potencializar a melhor correspondência determinada nas etapas anteriores. Neste caso, os mesmos padrões de busca empregados na Primeira Busca, mostrados na Figura 10, são aplicados novamente, com o ponto central atualizado com base na melhor correspondência identificada nas etapas anteriores. Aqui são permitidas apenas duas expansões e as novas posições devem permanecer dentro da área de busca definida (Gonçalves et al., 2022).

3.3 Estimação de Movimento *Affine*

A Estimação de Movimento *Affine* é uma ferramenta introduzida no padrão VVC, tendo a função de mapear movimentos não translacionais, tais como dimensionamento, rotação e cisalhamento, de forma a melhorar a eficiência de codificação (Maass et al., 2024). A AME realiza estes movimentos de acordo com os seus Pontos de Controle, do inglês *Control Points* (CP). Estes movimentos estão ilustrados na Figura 12.

Figura 12 – Tipos de movimentos da AME e seus respectivos Pontos de Controle.

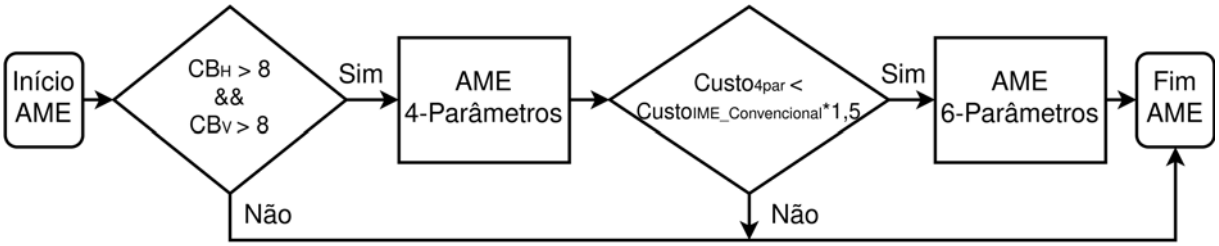


(a) Um Ponto de Controle. (b) Dois Pontos de Controle. (c) Três Pontos de Controle.

Fonte: Adaptada de Viana et al. (2024b).

O processo de execução da AME na predição Inter-Quadros do VVC está ilustrado na Figura 13, onde a AME é executada somente para CBs com dimensões maiores do que 8×8 (Maass et al., 2023).

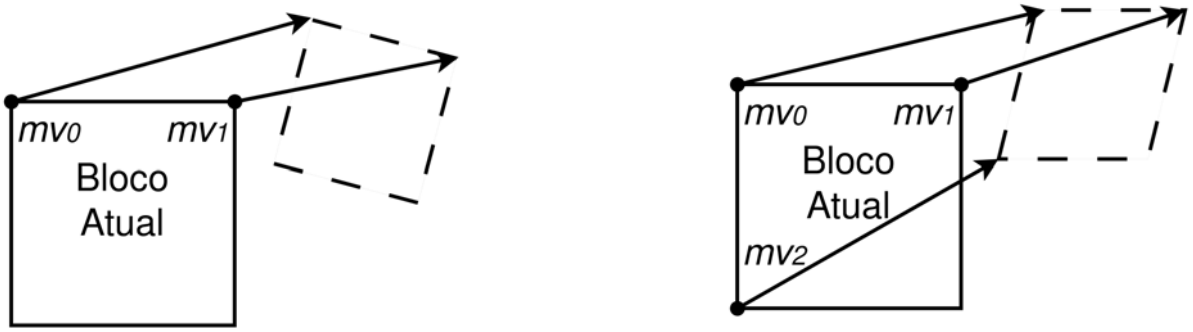
Figura 13 – Fluxograma das etapas da AME.



Fonte: Elaborada pelo autor.

Como mostrado na Figura 13, o VVC incorpora duas etapas de AME, chamadas de 4-Parâmetros e de 6-Parâmetros. O modelo AME de 4-Parâmetros é empregado para codificar movimentos mais simples, como escala e rotação. Por outro lado, o modelo AME de 6-Parâmetros é utilizado para movimentos mais complexos, abrangendo cortes e ajustes de proporção de aspecto (Muñoz et al., 2023). Consequentemente, dois ou três Vetores de Movimento são gerados (dependendo dos Pontos de Controle), localizados nos cantos superior-esquerdo, superior-direito e inferior-esquerdo (Storch; Palomino; Bampi, 2022), conforme ilustra a Figura 14.

Figura 14 – Modelos AME.



(a) 4-Parâmetros.

(b) 6-Parâmetros.

Fonte: Adaptada de Viana et al. (2024b).

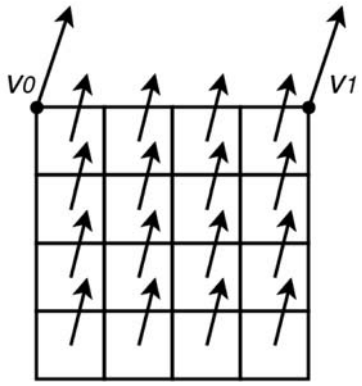
Para calcular a derivação dos Vetores de Movimento, são definidas duas equações considerando a localização por amostra (x, y) (Browne; Ye; Kim, 2022). Para a AME de 4-Parâmetros é utilizada a Equação (10), enquanto para a AME de 6-Parâmetros é utilizada a Equação (11). Onde (mv_{0x}, mv_{0y}) é o Vetor de Movimento do CP superior esquerdo (CP_0), (v_{1x}, mv_{1y}) é o MV do CP superior direito (CP_1) e (mv_{2x}, mv_{2y}) é o MV do CP inferior esquerdo (CP_2). H e W se referem à altura e à largura do bloco, respectivamente (Browne; Ye; Kim, 2022).

$$\begin{cases} mv_x = \frac{mv_{1x}-mv_{0x}}{W}x + \frac{mv_{0y}-mv_{1y}}{W}y + mv_{0x} \\ mv_y = \frac{mv_{1y}-mv_{0y}}{W}x + \frac{mv_{1x}-mv_{0x}}{W}y + mv_{0y} \end{cases} \tag{10}$$

$$\begin{cases} mv_x = \frac{mv_{1x}-mv_{0x}}{W}x + \frac{mv_{2x}-mv_{0x}}{H}y + mv_{0x} \\ mv_y = \frac{mv_{1y}-mv_{0y}}{W}x + \frac{mv_{2y}-mv_{0y}}{H}y + mv_{0y} \end{cases} \tag{11}$$

Assim como o TZS, o modo *Affine* é aplicável a 12 tamanhos de bloco, variando de 16×16 até 128×128 (Chen; Ye; Kim, 2020). Por simplificação, a AME no VVC não é baseada em amostras, mas em sub-blocos de luminância 4×4 (Viana et al., 2024b). O Vetor de Movimento da amostra central de cada sub-bloco é calculado de acordo com as equações anteriores e arredondado para a precisão 1/16, como ilustrado na Figura 15.

Figura 15 – Predição Affine por sub-blocos.



Fonte: Elaborada pelo autor.

Em seguida, na Compensação de Movimento, os filtros de interpolação são aplicados para gerar a predição de cada sub-bloco com base no Vetor de Movimento derivado. Os sub-blocos de crominância também têm o tamanho 4×4, sendo que o seu Vetor de Movimento é calculado a partir da média dos Vetores de Movimento dos sub-blocos de luminância superior esquerdo e inferior direito, na região co-localizada 8×8 de luminância. Isto ocorre devido à subamostragem 4:2:0 no vídeo de entrada.

3.4 Avaliação de Desempenho do VVC

Para avaliar o desempenho do VVC considerando propostas de alterações nas ferramentas do padrão, foram definidas Condições de Teste Comum, do inglês *Common Test Conditions* (CTCs) (Bossen et al., 2020). As CTCs definem especificações a serem utilizadas nos experimentos realizados, para manter um padrão entre trabalhos e, assim, permitir comparações justas entre eles. Entre as suas definições, estão incluídas uma lista de sequências de vídeo a serem utilizadas nos experimentos com o VVC. Estas sequências estão divididas em sete diferentes classes para organizar os vídeos, como mostrado na Tabela 2, que apresenta a classe, o nome, a resolução, a quantidade de quadros e o número de quadros por segundo para cada uma das sequências definidas nas CTCs.

Tabela 2 – Informações sobre as sequências de vídeo das CTCs.

Classe	Sequência	Resolução	Número de Quadros	Taxa de Quadros
A1	Tango2	3840×2160 (Ultra HD)	294	60
	FoodMarket4	3840×2160 (Ultra HD)	300	60
	Campfire	3840×2160 (Ultra HD)	300	30
A2	CatRobot	3840×2160 (Ultra HD)	300	60
	DaylightRoad2	3840×2160 (Ultra HD)	300	60
	ParkRunning3	3840×2160 (Ultra HD)	300	50
B	MarketPlace	1920×1080 (Full HD)	600	60
	RitualDance	1920×1080 (Full HD)	600	60
	Cactus	1920×1080 (Full HD)	500	50
	BasketballDrive	1920×1080 (Full HD)	500	50
	BQTerrace	1920×1080 (Full HD)	600	60
C	BasketballDrill	830×480 (WVGA)	500	50
	BQMall	830×480 (WVGA)	600	60
	PartyScene	830×480 (WVGA)	500	50
	RaceHorsesC	830×480 (WVGA)	300	30
D	BasketballPass	376×240 (WQVGA)	500	50
	BQSquare	376×240 (WQVGA)	600	60
	BlowingBubbles	376×240 (WQVGA)	500	50
	RaceHorses	376×240 (WQVGA)	300	30
E	FourPeople	1280×720 (HD)	600	60
	Johnny	1280×720 (HD)	600	60
	KristenAndSara	1280×720 (HD)	600	60
F	BasketballDrillText	830×480 (WVGA)	500	50
	ArenaOfValor	1920×1080 (Full HD)	600	60
	SlideEditing	1280×720 (HD)	300	30
	SlideShow	1280×720 (HD)	500	20

As CTCs também definem que os experimentos devem ser realizados considerando quatro valores para o Parâmetro de Quantização: 22, 27, 32 e 37. Estes valores variados de QP permitem a execução do teste sobre diferentes perfis de taxa de bits e qualidade e, assim, conduzem a uma avaliação mais completa das modificações propostas para o VTM. Deste modo, todos os experimentos que seguem as CTCs devem codificar quatro vezes cada uma das 26 sequências de vídeo acima, uma para cada QP. Além disso, esse conjunto de execuções deve ser realizado tanto para o codificador modificado quanto para o codificador âncora, para permitir comparações.

Por fim, as CTCs definem três configurações que podem ser usadas no VTM, visando diferentes aplicações. Estas configurações são: *All Intra*, *Low Delay* e *Random Access* (Choi, 2022).

- ***All Intra*:** Todos os quadros do vídeo são codificados somente pela Predição Intra-Quadro, ignorando a Predição Inter-Quadros.

- ***Low Delay*:** O primeiro quadro do Grupo de Quadros, do inglês *Group of Pictures* (GOP), é codificado totalmente com a Predição Intra-Quadro e os demais quadros podem ser codificados de forma conjunta com a Predição Inter-Quadros ou Intra-Quadro, mas sem usar a predição bidirecional. Essa configuração é utili-

zada em aplicações que demandam codificação em tempo real, com os quadros codificados na mesma ordem de exibição.

- ***Random Access*:** Se diferencia do *Low Delay* por permitir o uso da predição bidirecional e também por permitir que a ordem de codificação dos quadros seja não-linear. Novamente, o primeiro quadro do GOP é codificado totalmente com a Predição Intra-Quadro e os demais quadros podem ser codificados de forma conjunta com a Predição Inter-Quadros ou Intra-Quadro e a predição bidirecional pode ser usada. Esta configuração é a que atinge a maior eficiência de codificação,

As CTCs também definem o uso das métricas *Bjøntegaard Delta-Bitrate* (BD-BR) e/ou *Bjøntegaard Delta-PSNR* (BD-PSNR) (Bjontegaard, 2001) para comparar a eficiência de codificação de uma versão modificada do codificador em relação ao codificador âncora. De forma simplificada, é possível afirmar que o BD-BR calcula a variação na taxa de bits quando uma mesma qualidade objetiva de vídeo é definida tanto para o codificador modificado quanto para o codificador âncora. De forma oposta, o BD-PSNR calcula a variação na qualidade (em PSNR) quando uma mesma taxa de bits é definida tanto para o codificador modificado quanto para o codificador âncora. Valores positivos de BD-BR indicam um aumento na taxa de bits para a mesma qualidade visual, o que, consequentemente, representa uma perda na eficiência da codificação. Por outro lado, valores positivos de BD-PSNR indicam um aumento na qualidade para a mesma taxa de bits, o que indica um ganho na eficiência da codificação (Duarte, 2021). O BD-BR é expresso em percentual, enquanto o BD-PSNR é expresso em decibéis. Como ambas as métricas trazem informações semelhantes, esta dissertação irá utilizar apenas a métrica BD-BR, que é a métrica mais usada nos trabalhos da área.

4 APRENDIZADO DE MÁQUINA

Neste capítulo são apresentados os conceitos básicos relacionados ao Aprendizado de Máquina, com foco nas Árvores de Decisão, que é o método explorado neste trabalho.

Atualmente, estão sendo utilizadas diversas abordagens em propostas de redução de complexidade na codificação de vídeo, e uma das mais utilizadas nos últimos anos é o uso de Aprendizado de Máquina, devido à popularização da área e aos resultados expressivos que são possíveis ao usar ML. O Aprendizado de Máquina tornou-se uma das principais ferramentas para melhorar e otimizar codificadores de vídeo, dada a grande quantidade de dados que podem ser gerados e utilizados para treinar modelos de ML capazes de tomar decisões diversas no processo de codificação (Benjak et al., 2021).

O Aprendizado de Máquina é realizado através da extração dos dados considerados importantes para serem utilizados para o treinamento do modelo escolhido para ser utilizado. Estes dados são chamados de *features*, ou atributos (Burkov, 2019).

Existem três principais métodos de Aprendizado de Máquina: Aprendizado Supervisionado, Aprendizado Não Supervisionado e Aprendizado por Reforço (Somvanshi et al., 2016).

- **Aprendizado Supervisionado:** Envolve treinar um modelo usando dados previamente rotulados, para que cada informação de entrada tenha uma informação de saída. Esse tipo de Aprendizado de Máquina tem o objetivo de ensinar o modelo a reconhecer padrões nas *features* de entrada que ajudem a prever quais serão suas saídas.

- **Aprendizado Não Supervisionado:** É usado para encontrar padrões escondidos nas variáveis de entradas, pois este não recebe instâncias rotuladas como no aprendizado supervisionado, sendo então treinado para agrupar essas entradas com base em semelhanças e diferenças.

- **Aprendizado por Reforço:** Apenas recebe instâncias de entrada, mas o modelo é treinado sendo recompensado ou punido pelo ambiente baseado em suas escolhas, melhorando assim a cada novo treinamento.

Para validar os modelos de Aprendizado de Máquina, são utilizadas métricas que demonstram o desempenho dos modelos durante o treinamento. Por meio delas, é possível analisar se o modelo está alcançando o resultado desejado, sendo que, quanto mais próximo de 1 for o resultado da métrica, melhor. Algumas dessas métricas mais usadas incluem Precisão, Acurácia, *Recall* e *F1-Score* (Boukhatem; Youssef; Nassif, 2022).

- **Precisão:** É calculada como a razão entre os casos positivos da aplicação real e o total de previsões positivas geradas pelo modelo.
- **Acurácia:** É obtida através da divisão das predições corretas (positivas e negativas) e o total das predições realizadas pelo modelo.
- ***Recall*:** É obtida pela razão entre os verdadeiros positivos preditos pelo modelos e o total de casos positivos na aplicação real.
- ***F1-Score*:** É calculado como a média harmônica entre Precisão e *Recall*.

Existem diferentes modelos de Aprendizado de Máquina disponíveis para serem utilizados, como Árvore de Decisão, Floresta Aleatória, do inglês *Random Forest* (RF), e Rede Neural Convolucional, do inglês *Convolutional Neural Network* (CNN), entre outros. As CNNs e outros modelos de Aprendizado de Máquina baseados em redes neurais são uma subclasse chamada de Aprendizado Profundo. O modelo de Aprendizado de Máquina mais simples, rápido e fácil de implementar é a Árvore de Decisão, por ser fácil de entender e simples de implementar em *hardware*, tornando-a um alicerce básico para a ciência de dados (Prasad; Kumar; Mm, 2013).

4.1 Árvores de Decisão

A Árvore de Decisão é um classificador representado como uma divisão recursiva do espaço de instâncias, de forma que é um mapa ou grafo utilizado para avaliar o curso de ação resultando em uma predição probabilística (Papageorgiou; Stylios; Groumpos, 2006).

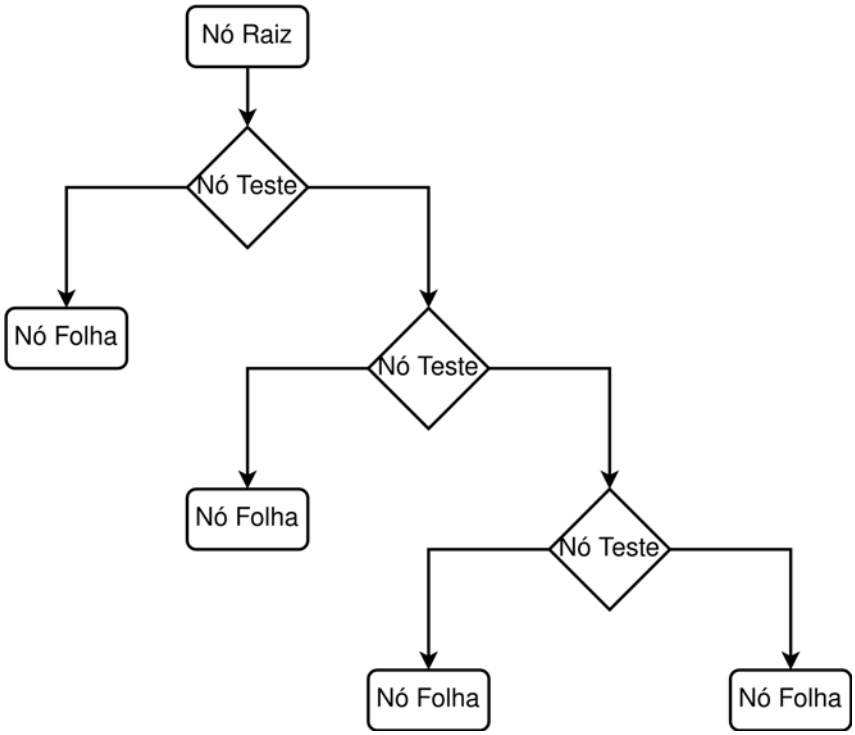
As Árvores de Decisão são compostas por três tipos de nós que formam uma árvore enraizada, estes sendo: raiz, teste e folha (Gupta et al., 2022).

- **Nó Raiz:** É o primeiro nó da Árvore de Decisão, possuindo somente arestas de saída e indica o início da classificação de cada instância.

- **Nó Teste:** Possui arestas de entrada e saída, dividindo o conjunto de instâncias em dois ou mais subconjuntos. Essa divisão ocorre após os algoritmos processarem os valores das variáveis, relacionando variáveis de entradas com as variáveis de saída de cada instância, determinando a melhor forma de separação.
- **Nó Folha:** Possui apenas arestas de entrada e é representado pelos nós mais profundos da árvore, contendo o valor da variável classe e indicando qual o resultado final da predição para aquele caso. Cada nó folha de uma Árvore de Decisão é associado a uma classe que representa o valor alvo mais apropriado. Por outro lado, o nó folha pode conter um vetor de probabilidade indicando a probabilidade do atributo alvo ter um determinado valor.

As instâncias são classificadas navegando desde o nó raiz da árvore até um nó folha, seguindo o resultado dos testes ao longo do caminho (Maimon; Rokach, 2005). A Figura 16 mostra a representação genérica de uma Árvore de Decisão com os seus nós raiz, teste e folha.

Figura 16 – Fluxograma de uma Árvore de Decisão genérica.



Fonte: Elaborada pelo autor.

As principais vantagens das Árvores de Decisão são sua interpretabilidade e a facilidade de derivar regras de decisão compreensíveis e flexíveis (ou seja, não fixas). Em uma Árvore de Decisão, a profundidade da árvore determina apenas o número máximo de condições que são utilizadas nas regras de decisão (Papageorgiou; Stylios; Groumpos, 2006).

Na área de Aprendizado de Máquina foram desenvolvidos diversos métodos para avaliar o grau de impureza de um conjunto de instâncias e, por consequência, sua relevância para o modelo. Os métodos mais utilizados para as Árvores de Decisão são o Índice Gini e a Entropia, sendo que ambos são baseados na impureza dos dados e buscam identificar as variáveis de entrada que possuem a maior relação com a variável de saída, contribuindo para a maior eficiência do modelo de classificação (Suthaharan, 2016).

- **Índice Gini:** É uma métrica que mede a frequência com que uma variável aleatória do conjunto de instâncias será rotulada erroneamente. Os valores possíveis para o Índice Gini estão entre 0 e 0,5, sendo que, quanto mais próximo de zero for o valor do índice, melhor, pois indica que a variável é mais homogênea (ou pura) (Suthaharan, 2016). O Índice Gini pode ser calculado através da Equação (12), em que p_j é a probabilidade da classe j.

$$Gini = 1 - \sum_{j} p_j^2 \tag{12}$$

- **Entropia:** É uma métrica que indica a desordem entre as variáveis de entrada em relação à classe. Assim como o Índice Gini, a melhor divisão é determinada pela variável com menor entropia. Os valores possíveis variam de 0 a 1. Quanto mais próximo de zero for o valor, menor é a entropia e mais homogêneas são as variáveis, indicando que a variável tende a identificar melhor uma determinada classe (Suthaharan, 2016). A Entropia pode ser calculada através da Equação (13).

$$Entropia = -\sum_{j} p_j \cdot log_2 \cdot p_j \tag{13}$$

Além do Índice Gini e da Entropia, é possível definir hiperparâmetros para melhorar uma Árvore de Decisão, como Mínimo de Amostras Divididas, do inglês *Min Samples Split*, Folha de Amostras Mínimas, do inglês *Min Samples Leaf*, Máximo de *Features*, do inglês *Max Features*, Profundidade Máxima, do inglês *Max Depth* e Máximo de Nós Folha, do inglês *Max Leaf Nodes*.

- ***Min Samples Split:*** Define o número mínimo de exemplos que um nó deve possuir para que seja possível realizar um particionamento, desta forma, controlando a altura da Árvore de Decisão.
- ***Min Samples Leaf:*** Define o número mínimo de exemplos que um nó folha deve possuir após um particionamento. Assim como o *Min Samples Split*, ele também controla a altura da Árvore de Decisão.
- ***Max Features:*** Define a quantidade de *features* que são avaliadas para decidir qual é a *feature* que gera o melhor particionamento em um nó da Árvore de Decisão.
- ***Max Depth:*** Define a profundidade máxima da Árvore de Decisão, possuindo impacto tanto no *Min Samples Split* quanto no *Min Samples Leaf*, já que ainda que um particionamento seja permitido de acordo com estes hiperparâmetros, se o mesmo resultar em uma profundidade além do que a qual o hiperparâmetro *Max Depth* define, este não pode ser realizado.
- ***Max Leaf Nodes:*** Define a quantidade máxima de nós folha que a árvore pode possuir, sendo também outro hiperparâmetro com impacto na altura da Árvore de Decisão.

Através destes métodos de avaliação e definição adequada dos hiperparâmetros, é possível avaliar quando uma Árvore de Decisão treinada está adequada para ser utilizada ou se necessita ser refeita.

5 REVISÃO SISTEMÁTICA DA LITERATURA

Neste capítulo é descrito o processo de Revisão Sistemática da Literatura (RSL) realizado sobre o tema deste trabalho. Baseado em Petersen et al. (2008), foram seguidas as seguintes etapas:

1. Definição das questões de pesquisa;
2. Busca de trabalhos;
3. Seleção dos trabalhos;
4. Definição de classificação e extração de dados.

5.1 Questões de Pesquisa

Os objetivos específicos de uma RSL são representados através das questões de pesquisa. Esta é uma etapa fundamental, visto que a operacionalização das demais etapas acontecerá com base nesta etapa (Petersen et al., 2008). É importante que as questões de pesquisa sejam respondidas ao final do processo, para que o objetivo da RSL seja atingido. Especificamente, neste trabalho, foram definidas as seguintes questões de pesquisa:

1. Quais são os trabalhos de pesquisa que apresentam soluções de otimização para a Predição Inter-Quadros do VVC usando Aprendizado de Máquina?
2. Quais são os trabalhos de pesquisa que apresentam soluções de otimização para o *Test Zone Search* do VVC?
3. Quais são os trabalhos de pesquisa que apresentam soluções de otimização para a Estimação de Movimento *Affine* do VVC?
4. Quais são as principais estratégias de Aprendizado de Máquina utilizadas nas soluções de otimização para a Predição Inter-Quadros do VVC?

5. Quais são as oportunidades de pesquisa ainda abertas na Predição Inter-Quadros do padrão VVC?

Desta forma, ao serem respondidas, as questões de pesquisa possibilitarão um melhor entendimento do estado da arte referente às soluções para a predição Inter-Quadros do padrão VVC. Além disso, será possível identificar lacunas de pesquisas que ainda podem ser exploradas.

5.2 Busca por Trabalhos

Durante a busca por trabalhos, a pesquisa foi realizada através de palavras-chave de busca, inseridas no site *Google Scholar*, que fornece acesso indireto às bases de dados existentes. Nesta pesquisa, foram utilizadas somente as bases de dados do *Institute of Electrical and Electronics Engineers* (IEEE), *Springer*, *Elsevier* e *Society of Photo-Optical Instrumentation Engineers* (SPIE) por serem fontes confiáveis de trabalhos verificados.

A busca foi realizada no período de 13/11/2024 até 18/11/2024. Foram utilizadas três *strings* de acordo com os escopos dos trabalhos a serem encontrados: (i) soluções de otimização do VVC utilizando Aprendizado de Máquina (*string* A), (ii) otimizações do *Test Zone Search* do VVC (*string* B) e (iii) otimizações da Estimação de Movimento *Affine* no VVC (*string* C). As *strings* estão apresentadas abaixo.

A. (“VVC" OR “Versatile Video Coding" OR “H.266") AND (“Inter-Frame" OR “Interframe" OR “Inter Prediction" OR “Inter-Prediction" OR “Interprediction") AND (“Machine Learning" OR “Machine-Learning" OR “ML" OR “Deep Learning")

B. (“VVC" OR “Versatile Video Coding" OR “H.266") AND (“Test Zone Search" OR “TZS") AND (“Complexity Reduction" OR “Time Reduction" OR “Fast" OR “Optimization")

C. (“VVC" OR “Versatile Video Coding" OR “H.266") AND (“Affine" OR “AME") AND (“Complexity Reduction" OR “Time Reduction" OR “Fast" OR “Optimization")

Para todas as *strings* foi utilizado um período único de publicações, pois todas buscam trabalhos relacionados ao codificador VVC que foi lançado no ano de 2020. Desta forma, foi definido um período de busca de 2019 (existiam muitos trabalhos sobre o VVC ainda antes do lançamento oficial) até 2024. No total, foram encontrados 2.332 trabalhos, sendo 1.320 deles resultantes da *string* A, 48 da *string* B e 964 da *string* C. Neste resultado inicial estão incluídos artigos de periódicos e eventos, livros e capítulos de livros.

5.3 Seleção dos Trabalhos

Após a busca por trabalhos com o uso das três *strings* ter sido realizada, foi feita uma seleção desses trabalhos através da aplicação de critérios de inclusão e de exclusão. A Tabela 3 apresenta estes critérios, que foram definidos com o objetivo de melhor atender às questões de pesquisa.

Tabela 3 – Critérios de inclusão e exclusão de trabalhos.

Critérios de Inclusão	Critérios de Exclusão
Estudos com foco no padrão de codificador VVC.	Estudos focados em outros codificadores.
Estudos que apresentam soluções para o TZS.	Trabalhos que não estão disponíveis na íntegra.
Estudos que apresentam soluções para a AME.	Trabalhos focados em vídeos 360°, vídeos 3D, *screen content*, *light fields* ou *point clouds*.
Trabalhos escritos em inglês ou português.	Trabalhos focados na etapa de decodificação.
Trabalhos que utilizam Aprendizado de Máquina na Predição Inter-Quadros.	Trabalhos focados na Predição Intra-Quadro.
Trabalhos que visam reduzir a complexidade do codificador de vídeo.	Estudos que apresentam soluções em *hardware* ou paralelismo.
Pertencem às bases de dados da *IEEE*, *Springer*, *Elsevier* e *SPIE*.	Trabalhos que não apresentem otimizações de redução de complexidade.
	Trabalhos publicados pelo próprio autor.

Como apresentado na Tabela 3, nesta seleção de trabalhos não foram incluídos dois trabalhos publicados em que o autor da dissertação é o primeiro autor, estes sendo **Viana et al. (2024a)** e **Viana et al. (2024b)**.

Os critérios de inclusão e exclusão dos trabalhos foram aplicados em duas etapas de refinamento:

I. Identificação do local de publicação, leitura do título e do resumo;

II. Leitura da introdução, dos resultados e da conclusão.

Na Tabela 4 estão apresentados os dados quantitativos resultantes dessas etapas de refinamento.

Tabela 4 – Resultado da seleção de trabalhos.

***String* de Busca**	**Total Inicial**	**Total Refinamento I**	**Total Refinamento II**
A	1320	90	15
B	48	9	2
C	964	29	9
Total	2332	128	26

Como mostrado na Tabela 4, depois de realizar o processo de refinamento, foram selecionados 26 trabalhos no total, sendo 15 deles resultantes da *string* A, dois da *string* B e 9 da *string* C. Porém, como a *string* A abrange também alguns aspectos das *strings* B e C, alguns trabalhos se repetiram entre as *strings*. Contabilizando somente uma vez cada trabalho encontrado, restaram 23 trabalhos diferentes no total.

5.4 Classificação e Descrição dos Trabalhos

Os trabalhos encontrados e selecionados foram divididos em três tópicos, um para cada *string*, para melhor apresentar e comparar estes trabalhos.

- Otimizações da Predição Inter-Quadros do VVC com Aprendizado de Máquina;
- Otimizações do *Test Zone Search* no VVC;
- Otimizações da Estimação de Movimento *Affine* no VVC.

5.4.1 Otimizações da Predição Inter-Quadros do VVC com Aprendizado de Máquina

Nesta subseção estão os trabalhos encontrados e refinados referentes a otimizações na Predição Inter-Quadros do VVC utilizando métodos de Aprendizado de Máquina. Foram destacados qual versão do software de referência VTM foi utilizada, qual ferramenta da predição Inter foi modificada, qual o método de Aprendizado de Máquina (ML) foi utilizado e quais os valores resultantes da redução do tempo total de codificação (TR) e da perda de eficiência de codificação através do *Bjøntegaard Delta-Bitrate* (BD-BR) (Bjontegaard, 2001). Também foram incluídos trabalhos que utilizaram o software de referência *Fraunhofer Versatile Video Encoder* (VVenC), que é uma implementação rápida do VVC (JVET - Joint Video Experts Team, 2020). Os trabalhos e os seus respectivos dados estão apresentados na Tabela 5.

Tabela 5 – Trabalhos encontrados sobre Predição Inter-Quadros do VVC com Aprendizado de Máquina.

Trabalho	Versão VTM	Ferramenta Modificada	Método de ML	TR Médio	BD-BR Médio
Amestoy et al. (2019)	5.0	Particionamento QTBT	RF	61,50%	2,22%
Kulupana et al. (2021)	10.0	Particionamento QTMT	RF	41,00%	1,33%
Pan et al. (2021)	6.0	Particionamento QTMT	CNN	30,36%	3,00%
Duarte et al. (2022)	9.0	AME	RF	08,49%	0,18%
Li et al. (2022)	10.0	Particionamento QTMT	RF	15,27%	1,87%
Lindino et al. (2022)	9.0	Particionamento QTMT	RF	34,76%	1,03%
Shen et al. (2022)	7.0	Particionamento QTMT	DT	51,00%	1,65%
Tissier et al. (2022)	10.2	Particionamento QTMT	DT	31,80%	1,11%
Xie et al. (2022)	1.0.0 (VVenC)	Inter-Quadros Inteira	RF	07,71%	1,48%
Xu et al. (2022)	13.0	Particionamento QTMT	RF	37,30%	1,41%
Huang et al. (2023)	10.0	AME	DT	10,33%	0,14%
Sagrilo et al. (2023)	16.2	AME	RF	02,98%	0,07%
Lin et al. (2024)	13.0	Particionamento QTMT	SCLIPEst-Net	45,14%	2,40%
Taabane et al. (2024)	1.7.0 (VVenC)	Particionamento QTMT	LightGBM	43,91%	2,90%
Wei et al. (2024)	13.0	Particionamento QTMT	CNN	23.94%	1,51%

O trabalho **Amestoy et al. (2019)** propõe um esquema de particionamento de bloco *Quad Tree Binary Tree* (QTBT) leve e ajustável, baseado em uma abordagem de Aprendizado de Máquina com o uso de Florestas Aleatórias, obtendo uma redução do tempo médio de codificação entre 25% e 61%, perdendo de 0,4% até 2,2% de eficiência de codificação.

No *paper* **Kulupana; Kumar M; Blasi (2021)** é proposta uma abordagem de Aprendizado de Máquina, com o uso de Florestas Aleatórias, para prever as decisões ideais de particionamento *Quad Tree Multi-type Tree* (QTMT), antes de codificar um bloco. Esse trabalho atingiu uma redução do tempo de codificação entre 32% e 59%, perdendo de 0,82% até 3,31% em eficiência de codificação.

No artigo **Pan et al. (2021)** é apresentado um método de codificação rápido baseado em Rede Neural Convolucional para encerrar antecipadamente o particionamento QTMT das CUs. Ele reduziu uma média de 30,63% do tempo de codificação, enquanto o BD-BR aumentou em cerca de 3%.

Duarte et al. (2022) apresenta uma proposta de otimização da Estimação de Movimento *Affine* utilizando Florestas Aleatórias para decidir quando é proveitoso que a AME seja executada durante a codificação. Ele conseguiu uma redução média do tempo total de codificação de 8,49% e uma redução média do tempo da *Affine* de 46,94%, com uma perda de eficiência de codificação de 0,18% em BD-BR.

O trabalho **Li; Zhang; Yang (2022)** propõe um algoritmo de particionamento QTMT para as Unidades de Codificação baseado em Florestas Aleatórias, de forma a decidir quando cada CU deve ser particionada ou não. Este algoritmo conseguiu uma redução média do tempo total de 15,27%, enquanto teve uma perda de eficiência de 1,87% em BD-BR.

A proposta apresentada por **Lindino et al. (2022)** propõe uma solução baseada em Aprendizado de Máquina para decisões rápidas de particionamento QTMT de blocos utilizando um conjunto de Florestas Aleatórias, de forma a decidir se partições verticais e horizontais são necessárias para cada bloco candidato. Esta solução proposta alcançou uma redução média do tempo de codificação de 34,76% ao custo de uma perda de eficiência de codificação de 1,03% em BD-BR.

O artigo **Shen; Yang; Wang (2022)** propõe uma estrutura de decisão de particionamento QTMT eficaz, usando Árvores de Decisão. Desta forma, o método atingiu uma redução média do tempo de codificação de 51%, com uma perda de eficiência de 1,65% em BD-BR.

No trabalho **Tissier et al. (2022)** é apresentado um método que utiliza Árvores de Decisão para melhorar o particionamento QTMT. Este método conseguiu reduzir o tempo de codificação em 31,8%, enquanto teve uma perda de 1,11% em BD-BR.

O artigo de **Xie et al. (2022)** apresenta um algoritmo de término antecipado para a predição Inter-Quadros baseada em Florestas Aleatórias, mas usando o VVenC. Os seus resultados experimentais demonstraram que, no modo *Random Access*, o tempo de codificação do VVenC foi reduzido em 7,71% em média, enquanto o BD-BR aumentou em 1,48%.

O trabalho em **Xu; He (2022)** propõe um algoritmo para otimizar o particionamento QTMT utilizando Florestas Aleatórias de forma a tornar o particionamento das CUs mais rápido ao predizer qual deve ser o tamanho a ser particionado. O algoritmo alcançou uma redução de tempo de 37,3%, porém teve uma perda de eficiência de 1,41% em BD-BR.

O trabalho em **Huang et al. (2023)** apresenta um algoritmo para reduzir o custo computacional da *Affine* com o uso de Árvores de Decisão. Este algoritmo atingiu 10,20% e 10,33% de economia de tempo com perda de 0,12% e 0,14% em BD-BR nas configurações *Low Delay B* e *Random Access*, respectivamente.

Em **Sagrilo et al. (2023)** é proposta uma redução de custo computacional da *Affine ME* utilizando Florestas Aleatórias para decidir quando os tamanhos de bloco não devem ser codificados pela *Affine ME*. Este trabalho conseguiu uma redução do tempo de execução da *Affine ME* de 20% e uma redução do tempo total de codificação de 2,89%, com uma perda de eficiência de codificação de 0,07%.

No trabalho em **Lin et al. (2024)** é apresentado um método de Particionamento Inter baseado em Aprendizado Contrastivo Supervisionado, do inglês *Supervised-Contrastive-Learning-based Inter Partitioning* (SCLIP). Os resultados experimentais demonstram que o método proposto alcançou uma média de 45,14% de economia de tempo com um aumento de 2,40% em BD-BR na configuração *Random Access*.

O trabalho **Taabane et al. (2024)** também está focado no software VVenC (e não no VTM) e utiliza classificadores binários *Light Gradient Boosting Machine* (LightGBM) para acelerar a partição de CUs em quadros inter na configuração *Random Access*. Os resultados experimentais mostram que este método reduz o tempo de execução do *preset* mais lento do VVenC em 43,21%, com apenas um leve aumento de 2,9% na taxa de bits de BD-BR.

Por fim, o trabalho em **Wei et al. (2024)** propõe um *framework* que extrai características de luminância, movimento, resíduos e informações de quantização a partir dos quadros de vídeo e, em seguida, realiza a fusão dessas características por meio de uma Rede Neural Convolucional para prever o tamanho mínimo de partição das CUs. O método proposto economiza entre 10,14% e 56,62% do tempo de codificação, com o aumento no BD-BR limitado a uma faixa de 0,31% a 6,70%.

5.4.2 Otimizações do *Test Zone Search* no VVC

Nesta subseção estão os trabalhos encontrados e refinados referentes a otimizações no *Test Zone Search* no VVC. Foram destacados qual versão do software de referência VTM foi utilizada, qual método de otimização foi utilizado e quais os valores resultantes da redução do tempo total de codificação (TR) e do *Bjøntegaard Delta-Bitrate* (BD-BR). Os trabalhos e os seus respectivos dados estão apresentados na Tabela 6.

Tabela 6 – Trabalhos encontrados sobre reduções de complexidade do *Test Zone Search* no VVC.

Trabalho	Versão VTM	Método Utilizado	TR Médio	BD-BR Médio
Sant'anna et al. (2021)	6.2	Análise Estatística	16,40%	0,37%
Yu; Yang (2023)	1.0.0 (VVenC)	Mudança no Padrão Busca	00,23%	0,12%

No trabalho **Sant'anna et al. (2021)** foi proposto um algoritmo que usa a influência que a taxa de bits do Vetor de Movimento causa nos padrões de busca da IME para podar regiões de pesquisa por meio de um critério simples e eficiente, que é aplicado por bloco. O método conseguiu reduzir o custo computacional da IME em aproximadamente 86,69%, com uma perda de eficiência de 0,74% em BD-BR.

No trabalho **Yu; Yang (2023)** foi proposto um algoritmo de otimização para o ponto de busca inicial e padrões de busca. Esta solução foi implementada no *software* VVenC, versão 1.0.0 e conseguiu reduzir o tempo de codificação em 0,23% com uma perda de eficiência de 0,12% em BD-BR.

5.4.3 Otimizações da Estimação de Movimento *Affine* no VVC

Nesta subseção estão os trabalhos encontrados e refinados referentes a otimizações na Estimação de Movimento *Affine* no VVC. Novamente, foram destacados qual versão do software de referência VTM foi utilizada, qual método de otimização foi utilizado e quais os valores resultantes da redução do tempo total de codificação (TR) e do *Bjøntegaard Delta-Bitrate* (BD-BR). Os trabalhos e os seus respectivos dados estão apresentados na Tabela 7.

Tabela 7 – Trabalhos encontrados sobre reduções de complexidade da Affine no VVC.

Trabalho	Versão VTM	Método Utilizado	TR Médio	BD-BR Médio
Park et al. (2019)	3.0	Análise Estatística	05,00%	0,10%
Guan et al. (2021)	6.0	Análise Estatística	15,50%	0,50%
Duarte et al. (2022)	9.0	Florestas Aleatórias	08,49%	0,18%
Loose et al. (2022)	14.0	Análise Estatística	18,85%	0,52%
Storch et al. 2022	12.0	Análise Estatística	27,58%	0,16%
Huang et al. (2023)	10.0	Árvores de Decisão	10,33%	0,14%
Pejman et al. (2023)	14.0	Análise Estatística	11,00%	0,82%
Sagrilo et al. (2023)	16.2	Florestas Aleatórias	02,98%	0,07%
Sheng et al. (2024)	18.0	Resolução Rápida de Equações Lineares	05,70%	0,07%

O trabalho **Park; Kang (2019)** propõe um método de codificação rápida para facilitar um processo da *Affine ME* empregando *features* úteis para ignorar processos redundantes. Resultados experimentais mostram que o método proposto reduz o tempo da AME do VTM em 63% em média, enquanto a perda de codificação fica em cerca de 0,1%.

No trabalho **Guan; Sun (2021)** é proposta uma otimização da AME usando *features* que usam o modo *Affine Skip* para blocos que não exigem predição *Affine*. Nas sequências de teste das classes C e D, o tempo de codificação foi economizado na média em 15,50%, resultando em um aumento médio no BD-RATE de 0,502%.

O trabalho **Loose et al. (2022)** propõe uma otimização da Predição Inter-Quadros realizando *early skips* nas predições Unidirecional, Bidirecional e *Affine* com base nos tamanhos de bloco que são executados menos vezes e que consomem muito tempo para serem executados. O trabalho conseguiu reduzir o tempo de codificação em 18,85% com uma perda de eficiência de 0,52% em BD-BR.

A proposta de **Storch; Palomino; Bampi (2022)** apresenta uma modelagem de predição *Affine* visando a implementação em GPU, extraindo o máximo de paralelismo de tais plataformas. Resultados experimentais mostram que, quando aplicado aos blocos 128$\times$128, o trabalho proposto pode acelerar a predição *Affine* em 57,21 vezes quando comparado a um codificador totalmente sequencial, com uma pequena penalidade na eficiência de codificação de 0,16% BD-BR.

O trabalho **Pejman et al. (2023)** propõe um novo método de decisão rápida ajustável para AME, em que ignora o processo AME para blocos com baixo custo de taxa-distorção, na estimação de movimento translacional, pois os autores mostram que ignorar esses blocos reduz o tempo de codificação sem afetar significativamente o desempenho da compressão. Para ignorar o processo AME, um limite de custo é determinado usando o modelo de Regressão Linear Múltipla. Resultados experimentais mostram que, com o limite padrão, o método proposto pode reduzir o tempo de codificação do VTM em 8% em média, nas classes B, C e D, com BD-BR de 0,44%. Para as mesmas classes e utilizando 1,5 vezes o limite padrão, é possível chegar a 11% de redução de custo computacional com BD-BR de 0,82%

Por fim, o artigo **Sheng et al. (2024)** apresenta um algoritmo rápido para a resolução de equações lineares *Affine* e seu projeto de *hardware*. Este trabalho alcançou uma economia média de tempo de 5,3% e 5,7%, com um aumento de 0,03% e 0,07% na taxa de BD-BR nas configurações de *Low Delay P* e *Random Access*, respectivamente.

Os trabalhos **Duarte et al. (2022)**, **Huang et al. (2023)** e **Sagrilo et al. (2023)** já foram apresentados na seção 5.4.1 e não serão apresentados novamente, embora sigam na tabela para critério de melhor visualização.

5.5 Análise de Todos os Trabalhos Relacionados Encontrados

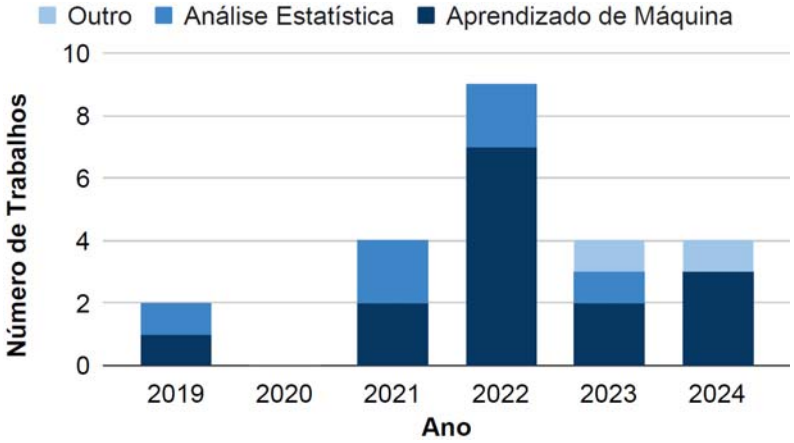
Para melhor comparar e analisar todos os trabalhos relacionados encontrados, os resultados encontrados para as três buscas foram incluídos na Tabela 8, contendo as suas versões do VTM, ferramenta que foi modificada, método utilizado para realizar a otimização e os seus resultados de redução de tempo total de codificação e BD-BR.

Tabela 8 – Trabalhos encontrados utilizando as três strings de busca.

Trabalho	Versão do VTM	Ferramenta Modificada	Método Utilizado	TR Médio	BD-BR Médio
Amestoy et al. (2019)	5.0	Particionamento QTBT	Florestas Aleatórias	61,50%	2,22%
Park et al. (2019)	3.0	AME	Análise Estatística	05,00%	0,10%
Guan et al. (2021)	6.0	AME	Análise Estatística	15,50%	0,50%
Kulupana et al. (2021)	10.0	Particionamento QTMT	Florestas Aleatórias	41,00%	1,33%
Pan et al. (2021)	6.0	Particionamento QTMT	Redes Neurais Convolucionais	30,36%	3,00%
Sant'anna et al. (2021)	6.2	TZS	Análise Estatística	16,40%	0,37%
Duarte et al. (2022)	9.0	AME	Florestas Aleatórias	08,49%	0,18%
Li et al. 2022	10.0	Particionamento QTMT	Florestas Aleatórias	15,27%	1,87%
Lindino et al. (2022)	9.0	Particionamento QTMT	Florestas Aleatórias	34,76%	1,03%
Loose et al. (2022)	14.0	AME	Análise Estatística	18,85%	0,52%
Shen et al. 2022	7.0	Particionamento QTMT	Árvores de Decisão	51,00%	1,65%
Storch et a. (2022)	12.0	AME	Análise Estatística	27,58%	0,16%
Tissier et al. (2022)	10.2	Particionamento QTMT	Árvores de Decisão	31,80%	1,11%
Xie et al. (2022)	1.0.0 (VVenC)	Inter-Quadros Inteira	Florestas Aleatórias	07,71%	1,48%
Xu; He (2022)	13.0	Particionamento QTMT	Florestas Aleatórias	37,30%	1,41%
Huang et al. (2023)	10.0	AME	Árvores de Decisão	10,33%	0,14%
Sagrilo et al. (2023)	16.2	AME	Florestas Aleatórias	02,98%	0,07%
Yu; Yang (2023)	1.0.0 (VVenC)	TZS	Mudança no Padrão Busca	00,23%	0,12%
Lin et al. (2024)	13.0	Particionamento QTMT	SCLIPEst-Net	45,14%	2,40%
Pejman et al. (2023)	14.0	AME	Análise Estatística	11,00%	0,82%
Sheng et al. (2024)	18.0	AME	Resolução Rápida de Equações Lineares	05,70%	0,07%
Taabane et al. (2024)	1.7.0 (VVenC)	Particionamento QTMT	LightGBM	43,91%	2,90%
Wei et al. (2024)	13.0	Particionamento QTMT	Redes Neurais Convolucionais	23.94%	1,51%

Para averiguar as técnicas utilizadas nos trabalhos apresentados na Tabela 8, foram analisados quais foram os tipos de métodos utilizados em cada um deles, separando em: Aprendizado de Máquina, Análise Estatística e Outro. Esta análise é mostrada na Figura 17.

Figura 17 – Número de trabalhos por ano de publicação e técnica utilizada.

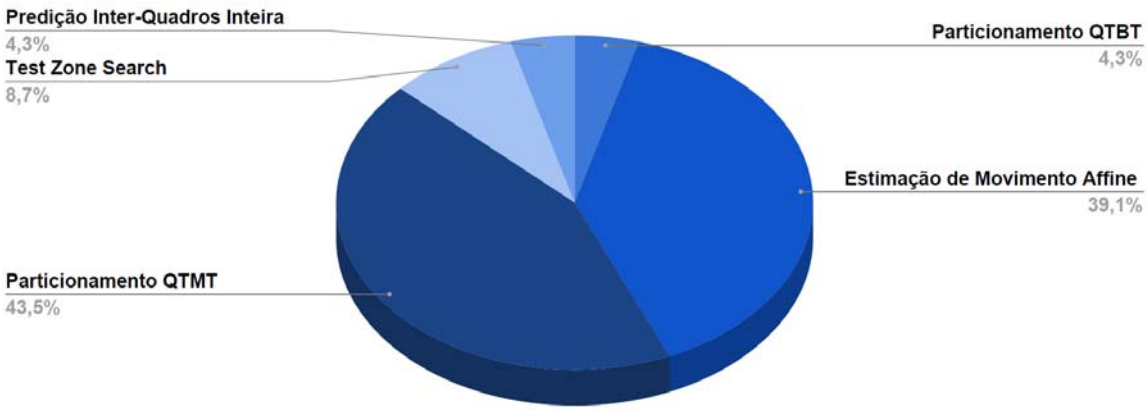


Fonte: Elaborada pelo autor.

Pela Figura 17, é possível concluir que o uso de técnicas de Aprendizado de Máquina tem sido cada vez mais usado. Além disso, é seguro afirmar que há uma tendência de soluções utilizando Aprendizado de Máquina no futuro próximo neste cenário de pesquisa. Considerando que existem vários modelos de Aprendizado de Máquina a serem explorados e uma grande quantidade de dados pode ser extraída do processo de codificação de vídeo, vale concluir que esse tipo de solução tende a continuar a crescer nos próximos anos. Do ponto de vista dos anos de publicação, 2022 foi o ano com maior número de trabalhos publicados, mas destaca-se que existe um aumento no número de publicações entre 2023 e 2024, indicando um provável reaquecimento da pesquisa nesta temática.

Também foi realizada uma análise de quantos trabalhos focam em cada uma das ferramentas da Predição Inter-Quadros do VVC mostrados na Tabela 8. Esta análise é apresentada na Figura 18.

Figura 18 – Porcentagem de trabalhos por ferramenta alvo.



Fonte: Elaborada pelo autor.

A Figura 18 mostra que o Particionamento QTMT e a Estimação de Movimento *Affine* são os destaques na maioria dos trabalhos. Destes, 10 trabalhos focaram no Particionamento QTMT e outros 10 focaram na AME, com os demais cinco trabalhos focando em outras ferramentas da Predição Inter-Quadros.

5.6 Considerações Finais

Através dos resultados apresentados neste capítulo, foi possível responder às cinco questões de pesquisa definidas nesta revisão sistemática:

1. Quais são os trabalhos de pesquisa que apresentam soluções de otimização para a Predição Inter-Quadros do VVC usando Aprendizado de Máquina?

   Os trabalhos publicados na literatura que apresentam soluções de otimização para a Predição Inter-Quadros do VVC usando Aprendizado de Máquina estão apresentados e descritos na Subseção 5.4.1. No total foram discutidos 15 trabalhos publicados na literatura, sendo que a maioria deles têm foco no particionamento do quadro, reduzindo o número de blocos avaliados. O método de Aprendizado de Máquina mais usado foi o RF.

2. Quais são os trabalhos de pesquisa que apresentam soluções de otimização para o *Test Zone Search* do VVC?

   Foram encontrados apenas dois trabalhos publicados com soluções de otimização para o TZS. Estes trabalhos estão apresentados e descritos na Subseção 5.4.2. Nenhum destes trabalhos utiliza Aprendizado de Máquina.

3. Quais são os trabalhos de pesquisa que apresentam soluções de otimização para a Estimação de Movimento *Affine* do VVC?

   Os trabalhos publicados com soluções de otimização para a Estimação de Movimento *Affine* do VVC foram apresentados e descritos na Subseção 5.4.3. No total, foram encontrados nove trabalhos com foco na redução do custo computacional da Affine e apenas três trabalhos usaram soluções baseadas em aprendizado de máquina.

4. Quais são as principais estratégias de Aprendizado de Máquina utilizadas nas soluções de otimização para a Predição Inter-Quadros do VVC?

   Analisando os trabalhos publicados, as principais estratégias de Aprendizado de Máquina utilizadas para reduzir o custo computacional da Predição Inter-Quadros do VVC foram Árvores de Decisão e Florestas Aleatórias, mas também existem trabalhos que utilizam Redes Neurais Convolucionais.

5. Quais são as oportunidades de pesquisa ainda abertas na Predição Inter-Quadros do padrão VVC?

   Apesar de existirem diversos trabalhos relacionados à Predição Inter-Quadros do VVC, ainda é possível realizar novos trabalhos nesta etapa, principalmente focando no *Test Zone Search* e na Estimação de Movimento *Affine*, pois ainda existem poucos trabalhos publicados nessas temáticas, e poucos destes trabalhos usam Aprendizado de Máquina para reduzir os custos computacionais destas ferramentas.

6 AVALIAÇÃO ESTATÍSTICA DA PREDIÇÃO INTER-QUADROS DO VVC

Neste capítulo é apresentada uma avaliação estatística das ferramentas da Predição Inter-Quadros, com destaque para as ferramentas alvo deste trabalho: o *Test Zone Search* e a Estimação de Movimento *Affine*.

6.1 Metodologia de Avaliação

Para realizar este trabalho, um conjunto de experimentos foi conduzido para melhor compreender os comportamentos do TZS e da AME durante o processo de codificação do VVC. Esses experimentos usaram o *software* de referência VTM versão 16.2 e consideraram as especificações das Condições Comuns de Teste (Bossen et al., 2020).

Neste trabalho, foram utilizadas as seguintes configurações no *software* VTM:

- Configuração *Random Access* (RA);
- Quatro valores de QP: 22, 27, 32 e 37;
- 23 sequências de vídeo diferentes agrupadas por classes: A1, A2, B, C, D e F, descritas na Tabela 2.

Nesses experimentos, foram utilizadas somente as sequências das classes A1, A2, B, C, D e F, pois são as definidas pelas CTCs para a configuração *Random Access*. Os primeiros 32 quadros de cada sequência foram utilizados, este sendo o tamanho do GOP utilizado na Predição Inter-Quadros com o *Random Access*. Não foram utilizados todos os quadros dos vídeos porque esses vídeos demoram muito tempo para serem codificados no modo *Random Access*, o que inviabilizaria a conclusão dos experimentos a tempo de serem relevantes para o trabalho desenvolvido nesta dissertação.

Todos os experimentos foram realizados em um servidor Intel Xeon CPU E5-2640 v3 @ 2,60 GHz, com oito núcleos e 32 GB de RAM.

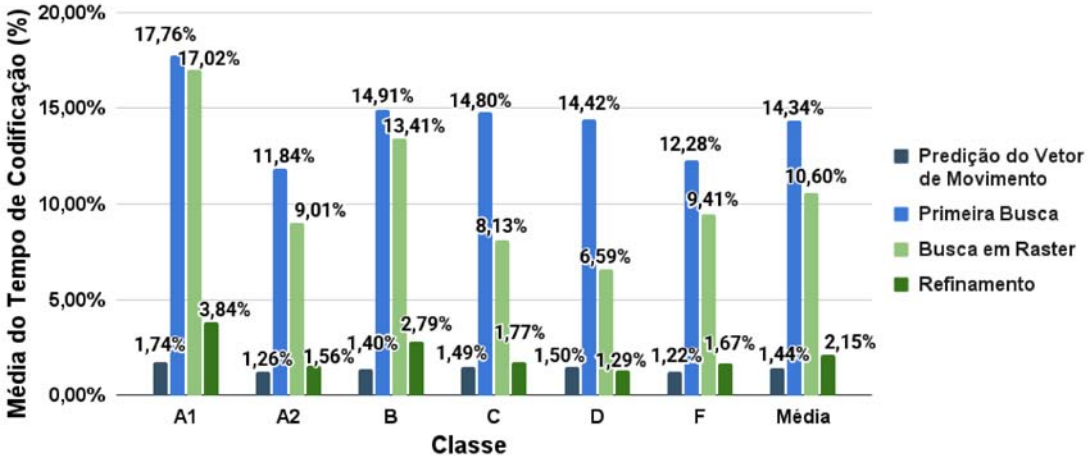
Para tornar a avaliação mais compreensível, ela foi dividida entre uma avaliação somente para o TZS, e outra somente para a AME. Em seguida, foi realizada uma nova avaliação que consistiu em sempre pular as três últimas etapas do TZS (a primeira etapa é mandatória) e a AME completa, de forma a analisar melhor os seus impactos no tempo total de codificação e na perda de eficiência de codificação. Esses últimos resultados são importantes para definir os limites possíveis para as soluções que foram desenvolvidas neste trabalho.

6.2 Avaliação do TZS

Os resultados do experimento com o TZS estão sumarizados em duas figuras, apresentadas mais abaixo. A Figura 19 ilustra as porcentagens de tempo de codificação de cada etapa do TZS em relação ao tempo total de codificação do VVC. Os dados consideram valores médios dos vídeos em cada uma das classes e, para cada vídeo, os resultados médios dos quatro QPs definidos nas CTCs. A Figura 20, por sua vez, apresenta o tempo médio consumido por cada etapa do TZS em relação ao tempo total de codificação do VVC para cada um dos quatro QPs definidos pelas CTCs. Novamente, foram considerados resultados médios, mas, neste caso, foram considerados todos os vídeos de todas as classes, para cada QP.

A primeira observação sobre esse experimento é que o TZS é uma ferramenta que consome parte significativa do tempo total de codificação do VVC. Em média, 28,52% do tempo de codificação é gasto na execução do TZS.

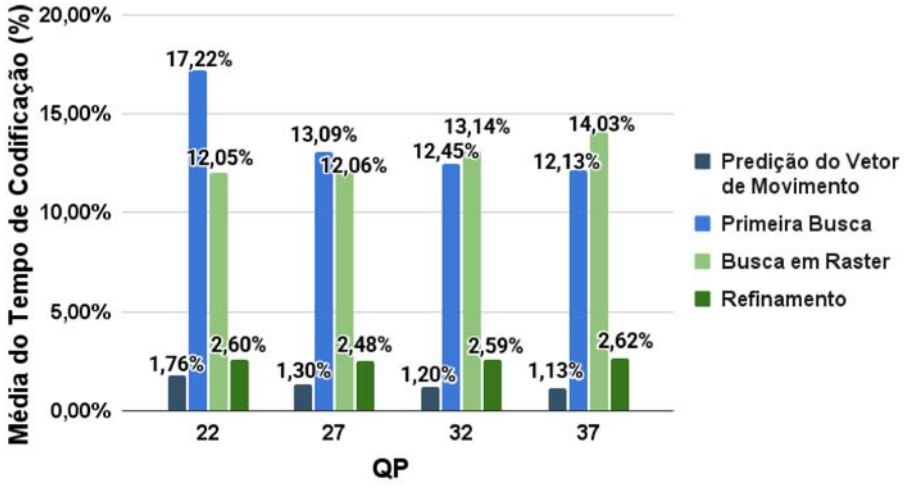
Figura 19 – Tempo médio utilizado por cada etapa do TZS em relação ao tempo total de codificação VVC, considerando diferentes classes de vídeo.



Fonte: Elaborada pelo autor.

Analisando a Figura 19 é possível observar que as etapas Primeira Busca e a Busca em *Raster* são as etapas com maior custo computacional em todas as classes de vídeos, utilizando, juntas, uma média de 24,93% do tempo total do codificador. Considerando resultados médios, após estas duas etapas, a etapa de Refinamento ocupa o terceiro maior percentual de tempo de execução, seguida pela ferramenta de Predição de Vetor de Movimento. Ainda analisando a Figura 19 é possível observar que na classe A1 é onde o TZS ocupa o maior percentual do tempo total de execução do VVC, com 40,35%. Por outro lado, na classe A2 é onde o TZS utiliza o menor percentual de tempo, com 23,67%. Esse resultado é importante, uma vez que ambas as classes possuem a mesma resolução. Ou seja, a variação no percentual é função de outras características dos vídeos que não a resolução em si.

Figura 20 – Tempo médio utilizado por cada etapa do TZS em relação ao tempo total de codificação VVC, considerando diferentes QPs.



Fonte: Elaborada pelo autor.

Analisando a Figura 20 é possível observar que o percentual de tempo total utilizado para executar o TZS é pouco afetado pelo valor de QP. Destaque para o QP menor (22), que é o QP em que o TZS ocupa o maior percentual de tempo total do codificador, com 33,63%.

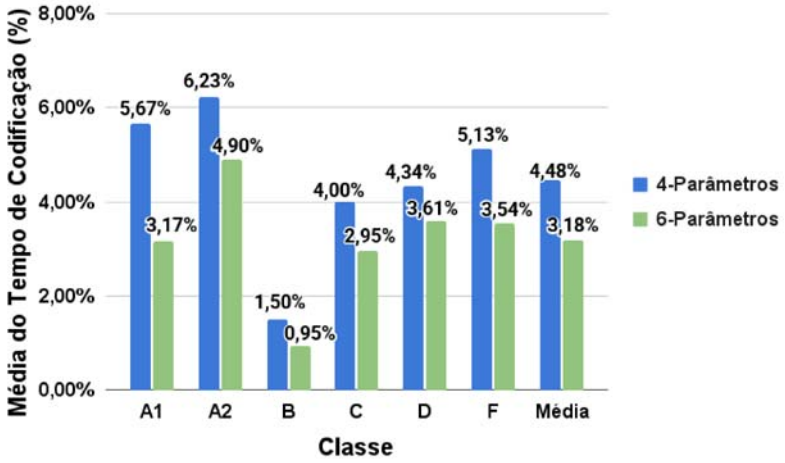
Por fim, os resultados nas Figuras 19 e 20 apontam que propostas para reduzir o custo computacional evitando as etapas de Primeira Busca, Busca em *Raster* e Refinamento têm potencial de gerar um impacto importante na redução de tempo de execução do codificador. Juntas, estas três etapas ocupam, na média, 27,09% do tempo total do codificador VVC.

6.3 Avaliação da AME

Uma avaliação similar à realizada para o TZS foi realizada para a AME. A Figura 21 mostra as porcentagens do tempo consumido por cada etapa da AME em relação ao tempo total de codificação VVC, para cada classe de vídeo. A Figura 22 ilustra o tempo médio consumido em cada etapa da AME em relação ao tempo total de codificação VVC para cada QP.

Com esse experimento foi possível perceber que a AME utiliza, em média, 7,66% do tempo total de codificação.

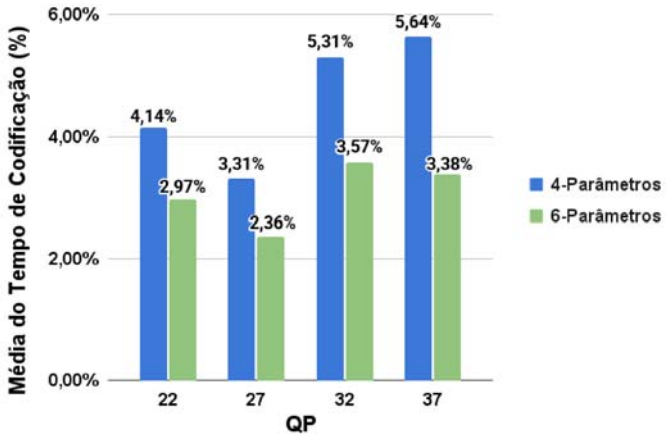
Figura 21 – Tempo médio utilizado por cada etapa da AME em relação ao tempo total de codificação VVC, considerando diferentes classes de vídeo.



Fonte: Elaborada pelo autor.

Analisando a Figura 21 é possível observar que a AME de 4-Parâmetros usa um maior percentual de tempo do codificador do que a AME de 6-Parâmetros. Esse resultado era esperado, pois, mesmo com operações mais complexas, a AME de 6-Parâmetros é realizada após a AME de 4-Parâmetros e, em muitos casos, ela nem é avaliada, como já discutido. Outra observação é que existe uma variação importante no percentual de tempo total utilizado pela ferramenta de acordo com a classe de vídeo. Os percentuais mais elevados estão nas classes A2 e A1, com 11,13% e 8,84% respectivamente, enquanto o percentual mais baixo está na classe B, com 2,45%.

Figura 22 – Tempo médio utilizado por cada etapa da AME em relação ao tempo total de codificação VVC, considerando diferentes QPs.



Fonte: Elaborada pelo autor.

Os resultados na Figura 22 indicam que a AME ocupa o maior percentual de tempo total de codificação quando QPs mais elevados são considerados.

Mesmo com um percentual de tempo total de execução bem inferior ao TZS, a AME pode ser completamente desligada no processo de codificação, implicando em ganhos importantes no tempo de execução.

6.4 Avaliação da Exclusão das Ferramentas TZS e AME no VVC

As últimas avaliações realizadas em relação ao TZS e à AME foram executar o VTM excluindo estas ferramentas para analisar o impacto que elas causam ao serem ignoradas. Tendo em vista que a primeira etapa do TZS, Predição do Vetor de Movimento, sempre precisa ser executada, ela foi mantida no VTM. Mas as três últimas etapas do TZS (Primeira Busca, Busca em *Raster* e Refinamento) foram ignoradas, enquanto a AME foi ignorada por completo.

Assim como descrito na Seção 3.4 deste trabalho, as perdas de eficiência de codificação foram avaliadas com a métrica BD-BR. Mas neste ponto, também foi avaliado o ganho no tempo de codificação. Nesse caso, o ganho foi medido com a métrica de Redução de Tempo (TR), refletindo as economias gerais no tempo de codificação ao comparar o VTM 16.2 modificado ao VTM 16.2 original e não modificado. O TR é calculado pela Equação (14), em que T_{VTM} é o tempo de codificação do VTM âncora, T_{VTM_Mod} é o tempo de codificação do VTM modificado e QP_i indica o i-ésimo valor de QP atualmente utilizado (Chen et al., 2022).

$$TR = \frac{1}{4} \sum_{QP_i \epsilon \{22,27,32,37\}} \frac{T_{VTM}(QP_i) - T_{VTM\_Mod}(QP_i)}{T_{VTM}(QP_i)} \tag{14}$$

Além do TR geral, também foi calculado o ganho de tempo no TZS (TR_{TZS}), já que esta ferramenta não foi completamente removida. Então o ganho de tempo está apresentado na Equação (15), em que T_{TZS} é o tempo de execução do TZS no VTM âncora, T_{TZS_Mod} é o tempo de execução do TZS no VTM modificado.

$$TR_{TZS} = \frac{1}{4} \sum_{QP_i \epsilon \{22,27,32,37\}} \frac{T_{TZS}(QP_i) - T_{TZS\_Mod}(QP_i)}{T_{TZS}(QP_i)} \tag{15}$$

A Tabela 9 apresenta os resultados ds avaliações de ganho de tempo de execução e perdas em eficiência de codificação, ao pular somente as três últimas etapas do TZS em todas as ocasiões.

Tabela 9 – Resultados de redução de tempo e eficiência de codificação com o VTM modificado para ignorar as três últimas etapas do TZS em todas as ocasiões.

Classe	Sequência	TR	TR_{TZS}	BD-BR
A1	Tango2	24,46%	96,21%	2,92%
	FoodMarket4	19,48%	96,57%	1,03%
	Campfire	32,77%	95,28%	0,86%
	Média Classe A1	25,57%	96,02%	1,60%
A2	CatRobot	19,37%	94,69%	4,07%
	DaylightRoad2	19,64%	94,65%	7,39%
	ParkRunning3	23,47%	93,26%	2,31%
	Média Classe A2	20,83%	94,20%	4,59%
B	MarketPlace	29,95%	94,83%	3,64%
	RitualDance	33,59%	95,50%	3,33%
	Cactus	22,81%	94,17%	2,57%
	BasketballDrive	24,54%	95,32%	3,20%
	BQTerrace	12,74%	90,42%	0,63%
	Média Classe B	24,73%	94,05%	2,67%
C	BasketballDrill	22,77%	93,60%	3,16%
	BQMall	21,02%	92,99%	3,76%
	PartyScene	24,09%	92,47%	1,35%
	RaceHorsesC	23,70%	93,44%	6,94%
	Média Classe C	22,89%	93,12%	3,80%
D	BasketballPass	16,75%	91,83%	3,84%
	BQSquare	13,63%	89,83%	1,02%
	BlowingBubbles	19,67%	91,12%	2,03%
	RaceHorses	22,41%	92,35%	13,80%
	Média Classe D	18,12%	91,28%	5,17%
F	BasketballDrillText	23,06%	93,61%	3,48%
	ArenaOfValor	24,08%	94,08%	6,42%
	SlideEditing	-02,88%	90,19%	9,62%
	SlideShow	11,67%	96,41%	2,43%
	Média Classe F	13,98%	93,57%	5,49%
	Média Geral	**21,02%**	**93,71%**	**3,89%**

Observando os resultados da Tabela 9 é possível observar que as três últimas etapas do TZS possuem um impacto importante em termos de tempo total de codificação, utilizando 21,92% do tempo total, na média. Impacto significativo também pode ser observado na eficiência de codificação, com uma perda média de 3,89% em BD-BR. É interessante perceber que existe uma estabilidade maior nos resultados de tempo do que nos resultados de eficiência de codificação. Isso quer dizer que o TZS, embora consuma uma fatia de tempo semelhante em todas as sequências, contribui mais para a eficiência de codificação para algumas sequências do que para outras.

A Tabela 10 apresenta os resultados das avaliações de ganho de tempo de execução e perdas em eficiência de codificação, ao pular somente a AME completa em todas as ocasiões. O ganho de tempo na AME não foi calculado pois, como ela foi completamente removida, o ganho foi de 100%, naturalmente.

Tabela 10 – Resultados de redução de tempo e eficiência de codificação com o VTM modificado para ignorar a AME completa em todas as ocasiões.

Classe	Sequência	TR	BD-BR
A1	Tango2	12,45%	1,45%
	FoodMarket4	14,36%	-0,10%
	Campfire	04,52%	0,04%
	Média Classe A1	10,44%	0,47%
A2	CatRobot	12,67%	4,77%
	DaylightRoad2	15,85%	4,04%
	ParkRunning3	07,13%	2,13%
	Média Classe A2	11,88%	3,65%
B	MarketPlace	01,51%	0,38%
	RitualDance	01,57%	0,23%
	Cactus	10,09%	4,40%
	BasketballDrive	10,60%	2,01%
	BQTerrace	10,50%	0,19%
	Média Classe B	06,85%	1,44%
C	BasketballDrill	09,25%	-0,09%
	BQMall	07,52%	0,48%
	PartyScene	06,93%	0,95%
	RaceHorsesC	08,72%	0,68%
	Média Classe C	8,10%	0,50%
D	BasketballPass	11,13%	0,22%
	BQSquare	09,87%	1,14%
	BlowingBubbles	09,44%	0,89%
	RaceHorses	07,57%	0,49%
	Média Classe D	09,50%	0,68%
F	BasketballDrillText	10,15%	0,07%
	ArenaOfValor	10,96%	1,12%
	SlideEditing	08,82%	-0,32%
	SlideShow	07,08%	0,74%
	Média Classe F	09,25%	0,40%
	Média Geral	**09,34%**	**1,19%**

Os resultados da Tabela 10 mostram que a AME consome, em média, menos tempo de codificação do que o TZS, chegando a uma média de 9,34% entre todas as sequências. Além disso, a contribuição da AME para a eficiência de codificação também é menor, chegando a uma perda de 1,19% de BD-BR na média. Em alguns casos, a remoção da AME inclusive melhorou levemente a eficiência de codificação (números negativos na Tabela 10).

Por fim, a Tabela 11 apresenta os resultados das avaliações de ganho de tempo de execução e perdas na eficiência de codificação ao sempre pular as três últimas etapas do TZS e a AME completa. Assim como na Tabela 10, o ganho de tempo na AME não é apresentado, pois ele é sempre de 100%. Essa análise é a mais relevante, pois demonstra o impacto conjunto do TZS e da AME quando ambas são desativadas.

Tabela 11 – Resultados de redução de tempo e eficiência de codificação com o VTM modificado para ignorar as três últimas etapas do TZS e a AME completa em todas as ocasiões.

Classe	Sequência	TR	TR_{TZS}	BD-BR
A1	Tango2	38,53%	96,08%	08,69%
	FoodMarket4	35,45%	96,58%	04,29%
	Campfire	37,63%	95,16%	00.98%
	Média Classe A1	37,20%	95,94%	04,65%
A2	CatRobot	34,57%	94,16%	09.36%
	DaylightRoad2	34,41%	94,12%	23,20%
	ParkRunning3	32,48%	92,78%	05,36%
	Média Classe A2	33,82%	93,69%	12,64%
B	MarketPlace	32,13%	94,71%	04,09%
	RitualDance	35,67%	95,40%	03,92%
	Cactus	33,41%	93,69%	09,63%
	BasketballDrive	30,70%	94,96%	16,74%
	BQTerrace	24,41%	90,02%	00,98%
	Média Classe B	31,26%	93,76%	07,07%
C	BasketballDrill	31,24%	93,27%	07,00%
	BQMall	29,54%	92,74%	08,12%
	PartyScene	31,81%	92,04%	02,54%
	RaceHorsesC	31,03%	92,94%	14,12%
	Média Classe C	30,90%	92,75%	07,14%
D	BasketballPass	28,88%	91,45%	04,81%
	BQSquare	20,62%	88,95%	03,10%
	BlowingBubbles	28,70%	90,50%	02,67%
	RaceHorses	29,55%	91,80%	20,29%
	Média Classe D	26,94%	90,67%	07,72%
F	BasketballDrillText	32,16%	93,37%	06,30%
	ArenaOfValor	27,70%	93,38%	32,86%
	SlideEditing	-14,42%	83,43%	31,32%
	SlideShow	19,40%	96,13%	05,54%
	Média Classe F	16,21%	91,58%	19,01%
	Média Geral	**29,39%**	**93,06%**	**09,84%**

Analisando a Tabela 11 é possível observar, inicialmente, que as três últimas etapas do TZS e a AME são ferramentas que consomem um percentual importante do tempo de codificação, com uma média de 29,39% do tempo total. Além disso, estas ferramentas também são muito importantes para melhorar a eficiência de codificação. Sem elas, os vídeos perdem, em média, 9,84% da eficiência em BD-BR. Esta perda é elevada e, assim, soluções para reduzir o tempo total de codificação destas ferramentas devem apresentar perdas de eficiência bem menores do que esta. De fato, a solução ideal seria aquela que atinge um ganho de 29,39% na redução do tempo total, com nenhuma perda em eficiência de codificação. Naturalmente, estes resultados são impossíveis de serem obtidos, mas são fronteiras importantes para avaliar as soluções desenvolvidas.

Outra observação interessante na Tabela 11 é que os resultados médios de redução de tempo são similares às somas dos resultados médios apresentados na Tabela 10) e na Tabela 9. Por outro lado, esse resultado não se repete ao considerar a eficiência de codificação. De fato, ao desligar as duas ferramentas de forma simultânea, as perdas de codificação são bastante superiores (9,84%) à soma das perdas de quando cada uma das ferramentas é desligada de forma isolada (5,08%). Uma explicação para esse comportamento é que, ao desligar uma ferramenta, a outra passa a ser mais usada, justamente para compensar as perdas causadas ao desligar uma das ferramentas.

Ainda a partir da Tabela 11 é possível perceber que a redução de tempo é maior para resolução de vídeo mais elevadas (Classes A1 e A2). No TZS, esse comportamento é menos perceptível pois o ganho de tempo médio é sempre maior que 90% para todas as classes.

Esses resultados indicam que a relevância do TZS e da AME varia significativamente entre as diferentes sequências de vídeo, em função de suas características específicas.

7 MODELOS DESENVOLVIDOS PARA ACELERAR O CODIFICADOR VVC

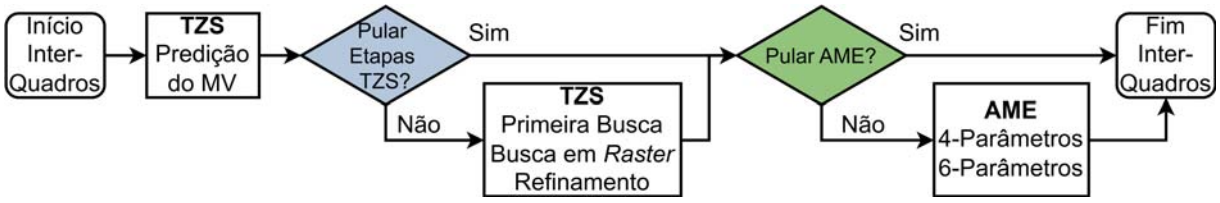
Neste capítulo são descritos os dois modelos de Aprendizado de Máquina desenvolvidos para reduzir o custo computacional da Predição Inter-Quadros do VVC através do uso de Aprendizado de Máquina.

Este trabalho propõe uma solução para acelerar o tempo total de codificação do codificador VVC, utilizando dois conjuntos de Árvores de Decisão para pular seletivamente duas das principais ferramentas da Predição Inter-Quadros do VVC: o *Test Zone Search* e a Estimação de Movimento *Affine*. Ele foi apresentado no artigo **Viana et al. (2025)**, publicado no *16th IEEE Latin American Symposium on Circuits and Systems* (LASCAS 2025).

Todo o processo de organização de *datasets* e treinamentos e testes das Árvores de Decisão foi feito em Python com a biblioteca *Scikit-Learn*, através da plataforma *Google Colab*.

Foram utilizados 12 modelos de Árvores de Decisão em cada uma das ferramentas, um modelo para cada tamanho de bloco suportado pelo TZS e pela AME. A Figura 23 ilustra como os modelos foram implementados no TZS, em azul, e na AME, em verde.

Figura 23 – Fluxograma da solução completa proposta para acelerar o TZS e a AME utilizando Árvores de Decisão.



Fonte: Adaptada de Viana et al. (2025).

Para garantir que os modelos sejam implementados corretamente, os modelos de Árvore de Decisão para o TZS foram inicialmente treinados e incorporados ao VTM. Em seguida, os modelos de DT para o AME foram treinados, levando em consideração a versão simplificada do TZS dentro do VTM. Por fim, os modelos de DT para o AME

foram integrados ao VTM para as avaliações finais.

7.1 Desenvolvimento das Árvores de Decisão para o TZS

A primeira etapa para desenvolver as Árvores de Decisão do TZS foi a extração de *features*. Para extrair *features* relevantes para as etapas de codificação que se deseja otimizar, o codificador VTM foi executado normalmente, evitando processamento adicional para a obtenção desses dados. As únicas alterações feitas no software do codificador foram para contar o tempo de execução das etapas específicas e as rotinas para extrações das *features*. O conjunto de sequências de vídeo utilizado para esta extração não possui nenhuma sequência das CTCs, que foram usadas nos experimentos já apresentados e que serão utilizadas na avaliação final dos resultados. Assim, foi possível evitar qualquer viés nos modelos desenvolvidos (Sagrilo et al., 2023).

A extração de *features* para a fase de treinamento utilizou um conjunto novo de 12 sequências de vídeo com diferentes resoluções:

- Quatro vídeos HD (*dark*, *KristenAndSara*, *Netflix_DinnerScene* e *Netflix_DrivingPOV*);
- Dois vídeos Full HD (*Netflix_TunnelFlag* e *rush_field_cuts*);
- Seis vídeos Ultra HD (*Beauty*, *BuildingHall2*, *Jockey*, *Lips*, *NetflixDancers* e *Netflix_ToddlerFountain*).

Já o teste das Árvores de Decisão utilizou um outro conjunto de três sequências de vídeo:

- Um vídeo HD (*Vidyo4*);
- Um vídeo Full HD (*touchdown*);
- Um vídeo Ultra HD (*SunBath*).

Esses vídeos estão disponíveis em três conjuntos de dados alternativos: UVG (Mercat; Viitanen; Vanne, 2020), NETVC (Daede; Norkin; Brailovskiy, 2019) e JVET (Boyce et al., 2018). Os primeiros 16 quadros de cada sequência foram codificados quatro vezes, uma para cada Parâmetro de Quantização (22, 27, 32 e 37).

As *features* foram salvas em tabelas *Comma-Separated Values* (CSV). Foram criadas quatro tabelas CSV de *features* para cada vídeo, uma para cada valor de QP, totalizando 48 tabelas geradas.

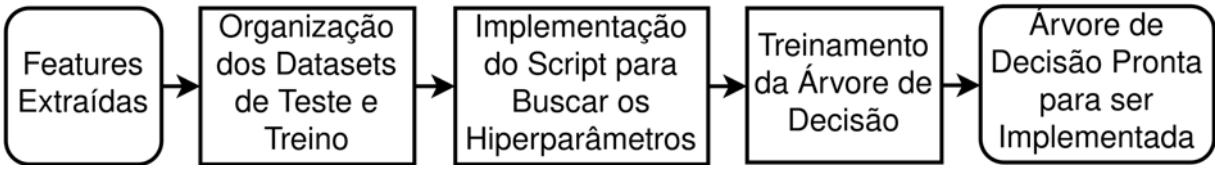
As *features* extraídas estão brevemente apresentadas na Tabela 12. Essas *features* foram escolhidas por dois motivos: (i) especificarem em qual ponto do vídeo e em qual ponto do quadro está ocorrendo a codificação, o que identifica os locais no quadro em que o TZS foi executado, de forma a identificar as posições em que locais ele deve ser executado ou não; (ii) serem variáveis utilizadas no processo de execução do TZS no código do VTM, sem exigir cálculos adicionais.

Tabela 12 – Descrição das features extraídas para o TZS.

Feature	Quantidade	Descrição
qp	1	Parâmetro de Quantização
depth	1	Profundidade do particionamento
qt_depth	1	Profundidade do particionamento QT
mt_depth	1	Profundidade do particionamento MT
cu_pos	2	Posição X e Y da CU dentro do quadro
mv_uni	4	Posição X e Y para o vetor de movimento gerado pela predição Unidirecional para Lista 0 e Lista 1
bits_mv_uni	2	Quantidade de bits gerado pelo MV na Lista 0 e Lista 1
mv_bi	4	Posição X e Y para o vetor de movimento gerado pela predição Bidirecional para Lista 0 e Lista 1
bits_mv_bi	2	Quantidade de bits gerado pela predição Bidirecional
smvd	1	Codificação MVD simétrica
atual_qp	1	QP atualmente sendo usado
rui_sad	1	SAD calculado dentro do MVP
cStruct_iBest	2	Melhor MV dentro do MVP

Utilizando as *features* descritas na Tabela 12 e que foram extraídas do VTM, foi realizado o processo de criação das 12 Árvores de Decisão a serem utilizadas no modelo de Aprendizado de Máquina para o TZS, sendo uma Árvore de Decisão para cada tamanho de bloco suportado pelo TZS. Este processo é ilustrado na Figura 24.

Figura 24 – Fluxograma do processo para criação e treinamento de uma Árvore de Decisão.



Fonte: Elaborada pelo autor.

Como apresentado na Figura 24, depois da extração das *features*, foram criados dois *datasets* de dados: um para o treinamento das Árvores de Decisão, e outro para o teste destas Árvores. Para garantir um processo de treinamento imparcial, esses *datasets* foram balanceados limitando seu tamanho a 100.000 elementos em cada *dataset*.

Estes dois *datasets* foram utilizados, inicialmente, em um *script* em Python para buscar os melhores hiperparâmetros para cada Árvore de Decisão (Duarte, 2021). Este *script* tomou por base um *script* desenvolvido no grupo, que foi adaptado para uso neste trabalho e está disponível no Apêndice A. O *script* utiliza o *Random Search* para realizar buscas com diferentes valores de hiperparâmetros com os *datsets* de treino e teste para, desta forma, encontrar os valores que geram os melhores resultados para a métrica *F1-Score* em diversas iterações (Bergstra; Bengio, 2012). Neste processo, foram utilizados os hiperparâmetros: *Criterion*, *Min Samples Split*, *Min Samples Leaf*, *Max Features*, *Max Depth* e *Max Leaf Nodes*. A Tabela 13 apresenta os espaços de busca definidos para cada hiperparâmetro.

Tabela 13 – Espaço de Busca para cada Hiperparâmetro no Random Search para as Árvores de Decisão.

Hiperparâmetro	Espaço de Busca
Criterion	[gini, entropy]
Min Samples Split	[2] $\bigcup$ [25, 500, passo 25]
Min Samples Leaf	[1] $\bigcup$ [10, 100, passo 10]
Max Features	[1] $\bigcup$ [1, 45, passo 1]
Max Depth	[1] $\bigcup$ [5, 100, passo 1]
Max Leaf Nodes	[1] $\bigcup$ [20, 700, passo 10]

Utilizando os espaços de busca mostrados na Tabela 13, o *script* realizou a busca dos melhores valores para cada hiperparâmetro utilizando 3.500 iterações. Os hiperparâmetros resultantes para cada uma das Árvores de Decisão para o modelo de Aprendizado de Máquina do TZS são apresentados na Tabela 14.

Tabela 14 – Hiperparâmetros calculados para cada DT do TZS.

Modelo	Criterion	Min Samples Split	Min Samples Leaf	Max Features	Max Depth	Max Leaf Nodes	F1-Score
DT TZS 16×16	entropy	002	60	25	87	450	0,92
DT TZS 16×32	gini	475	01	26	75	610	0,91
DT TZS 16×64	gini	150	80	26	59	410	0,91
DT TZS 32×16	entropy	325	70	23	79	440	0,91
DT TZS 32×32	entropy	225	20	21	57	700	0,90
DT TZS 32×64	entropy	150	70	25	95	460	0,89
DT TZS 64×16	entropy	300	30	25	53	650	0,92
DT TZS 64×32	entropy	425	10	25	37	270	0,90
DT TZS 64×64	entropy	002	20	25	41	550	0,90
DT TZS 64×128	entropy	150	60	26	73	470	0,87
DT TZS 128×64	entropy	125	01	23	65	700	0,87
DT TZS 128×128	entropy	225	40	25	83	230	0,88

Os hiperparâmetros encontrados foram utilizados para gerar as suas respectivas Árvores de Decisão utilizando os *datasets* de treino.

A Tabela 15 apresenta os resultados de Acurácia, Precisão, *Recall* e *F1-Score* para cada uma das Árvores de Decisão geradas para o modelo de Aprendizado de Máquina do TZS quando testadas com os *datasets* de teste.

Tabela 15 – Conjuntos de treinamento e resultados de testes para cada DT do TZS.

Modelo	Acurácia	Precisão	Recall	F1-Score
DT TZS 16×16	0,93	0,92	0,94	0,93
DT TZS 16×32	0,91	0,90	0,92	0,91
DT TZS 16×64	0,92	0,91	0,92	0,92
DT TZS 32×16	0,90	0,91	0,89	0,90
DT TZS 32×32	0,90	0,91	0,90	0,90
DT TZS 32×64	0,89	0,89	0,89	0,89
DT TZS 64×16	0,92	0,91	0,93	0,92
DT TZS 64×32	0,89	0,87	0,91	0,89
DT TZS 64×64	0,91	0,90	0,92	0,91
DT TZS 64×128	0,90	0,89	0,90	0,90
DT TZS 128×64	0,90	0,90	0,90	0,90
DT TZS 128×128	0,88	0,88	0,89	0,89

Analisando a Tabela 15, é possível observar que as Árvores de Decisão geradas para o modelo de Aprendizado de Máquina do TZS apresentam bons resultados por serem muito próximas de 1,0 e, portanto, estão adequadas para serem implementadas no *software* de referência VTM.

As 12 árvores geradas foram convertidas para C++ através da biblioteca *m2cgen* no mesmo *script* em Python utilizado em sua geração, disponível no Apêndice A. Assim, as árvores puderam ser implementadas diretamente no *software* VTM, cujo código foi escrito em C++.

Então as 12 DTs do TZS foram inseridas no VTM para decidir, para cada tamanho de bloco, se as três últimas etapas do TZS devem ser executadas ou não.

7.2 Desenvolvimento das Árvores de Decisão para a AME

Assim como foi feito no desenvolvimento das Árvores de Decisão para o TZS, o desenvolvimento dos modelos para a AME teve início com a extração das *features* do *software* VTM, nesse caso, as *features* relevantes para a AME. Esta extração foi realizada utilizando o VTM já com os 12 modelos do TZS implementados. Foram utilizadas as mesmas 12 sequências de vídeo para realizar a etapa de treinamento e as mesmas três sequências para o teste das DTs, como no processo do TZS. Além disso, novamente as sequências de vídeo foram codificadas normalmente no software VTM, somente modificado para extrair as *features* da AME em tabelas CSV. As *features* extraídas são as descritas na Tabela 16. Estas *features* foram escolhidas por dois motivos: (i) especificarem em qual ponto do vídeo e em qual ponto do quadro está

ocorrendo a codificação, o que identifica os locais no quadro em que AME foi executada, de forma a identificar as posições em que locais ele deve ser executado ou não; (ii) serem variáveis utilizadas no processo de execução da AME no código do VTM.

Tabela 16 – Descrição das features extraídas para a AME.

Feature	Quantidade	Descrição
qp	1	Parâmetro de Quantização
depth	1	Profundidade do particionamento
qt_depth	1	Profundidade do particionamento QT
mt_depth	1	Profundidade do particionamento MT
cu_pos	2	Posição X e Y da CU dentro do quadro
mv_uni	4	Posição X e Y para o vetor de movimento gerado pela predição Unidirecional para Lista 0 e Lista 1
bits_mv_uni	2	Quantidade de bits gerado pelo MV na Lista 0 e Lista 1
mv_bi	4	Posição X e Y para o vetor de movimento gerado pela predição Bidirecional para Lista 0 e Lista 1
bits_mv_bi	2	Quantidade de bits gerado pela predição Bidirecional
smvd	1	Codificação MVD simétrica
atual_qp	1	QP atualmente sendo usado
affine_pai	1	Indicador se o bloco pai é codificado com AME
custo_pai	1	Custo gerado pela predição do bloco pai
aff_viz_esq	1	Indicador se o vizinho a esquerda é codificado com AME
custo_viz_esq	1	Custo gerado pela predição do vizinho a esquerda
aff_viz_acima	1	Indicador se o vizinho acima é codificado com AME
custo_viz_acima	1	Custo gerado pela predição do vizinho acima
cu_atual_affine	1	Indicador se a CU atual possui o melhor modo de predição como AME
soma	1	Soma dos píxeis no bloco
media	1	Média dos valores dos píxeis no bloco
vari	1	Variância dos valores dos píxeis no bloco
grad	2	Gradientes horizontal e vertical do bloco
razao_grad	1	Razão entre os gradientes do bloco
razao_grad_píxeis	1	Razão entre os gradientes do pixel
desvioPadrao	1	Desvio padrão dos píxeis no bloco

Os passos seguintes foram similares aos utilizados na definição dos modelos do TZS. Novamente foram criados dois *datasets*: um de treino e outro de teste. Utilizando estes dois novos *datasets* no *script* em Python, foram calculados os melhores hiperparâmetros a serem utilizados nos modelos da AME, usando os mesmo espaços de busca para o *Random Search* utilizados para o modelo do TZS apresentados na Tabela 13.

Utilizando estes espaços de busca, os melhores valores para cada hiperparâmetro foram definidos pelo *script*, utilizando 3500 iterações. Os hiperparâmetros resultantes para cada uma das Árvores de Decisão para o modelo de Aprendizado de Máquina da AME são apresentados na Tabela 17.

Tabela 17 – Hiperparâmetros calculados para cada DT da AME.

Modelo	Criterion	Min Samples Split	Min Samples Leaf	Max Features	Max Depth	Max Leaf Nodes	F1-Score
DT AME 16×16	entropy	425	10	42	66	230	0,90
DT AME 16×32	entropy	075	20	44	09	020	0,89
DT AME 16×64	entropy	100	30	29	24	080	0,95
DT AME 32×16	entropy	450	60	45	96	020	0,90
DT AME 32×32	gini	100	30	44	08	570	0,91
DT AME 32×64	entropy	425	50	44	10	350	0,90
DT AME 64×16	entropy	175	70	34	11	650	0,95
DT AME 64×32	gini	250	01	44	22	130	0,90
DT AME 64×64	entropy	002	50	37	21	130	0,89
DT AME 64×128	entropy	350	40	29	51	090	0,94
DT AME 128×64	entropy	275	70	32	30	060	0,94
DT AME 128×128	gini	200	40	43	09	690	0,93

Usando estes hiperparâmetros, foram geradas as 12 Árvores de Decisão da AME com o *script* em Python disponível no Apêndice A. Estas Árvores de Decisão obtiveram os resultados de Acurácia, Precisão, *Recall* e *F1-Score* que estão apresentados na Tabela 18.

Tabela 18 – Conjuntos de treinamento e resultados de testes para cada DT da AME.

Modelo	Acurácia	Precisão	Recall	F1-Score
DT AME 16×16	0,91	0,88	0,95	0,91
DT AME 16×32	0,91	0,89	0,93	0,91
DT AME 16×64	0,93	0,92	0,94	0,93
DT AME 32×16	0,91	0,89	0,93	0,91
DT AME 32×32	0,91	0,89	0,92	0,91
DT AME 32×64	0,90	0,87	0,93	0,90
DT AME 64×16	0,93	0,91	0,95	0,93
DT AME 64×32	0,90	0,87	0,93	0,90
DT AME 64×64	0,87	0,86	0,89	0,88
DT AME 64×128	0,90	0,90	0,90	0,90
DT AME 128×64	0,90	0,91	0,89	0,90
DT AME 128×128	0,88	0,88	0,88	0,88

Observando os resultados da Tabela 18, é possível perceber que estas 12 Árvores de Decisão criadas para o modelo de Aprendizado de Máquina da AME possuem bons resultados, todas possuindo valores próximos de 1,0, assim como as Árvores do modelo do TZS. Desta forma, elas também foram consideradas adequadas para serem implementadas no software VTM.

As 12 árvores também foram convertidas para C++ e foram inseridas no software de referência VTM, que já possuía a aceleração do TZS com o seu conjunto de 12 Árvores de Decisão. No caso da AME, as Árvores de Decisão definem, para cada tamanho de bloco, se a AME deve ser executada ou não.

8 RESULTADOS

Neste capítulo, são apresentados e analisados os diferentes resultados obtidos com os modelos de Aprendizado de Máquina desenvolvidos, sendo usados em um cenário real de codificação de vídeos.

Para implementar e testar a solução proposta, foram realizados dois novos conjuntos de experimentos. Mais uma vez, esses experimentos utilizaram a versão 16.2 do VTM (Suehring, 2022) e aderiram às CTCs, utilizando a mesma configuração experimental apresentada no Capítulo 6:

- Configuração *Random Access* (RA);
- Quatro valores de QP: 22, 27, 32 e 37;
- 23 sequências de vídeo diferentes nas classes A1, A2, B, C, D e F.

Novamente, foram utilizados os primeiros 32 quadros de cada sequência de vídeo. É importante destacar que estas sequências das CTCs utilizadas para realizar a avaliação são completamente diferentes daquelas usadas nos processos de treinamento e validação.

Os experimentos foram conduzidos com cada vídeo sendo codificado em um único núcleo de processador para garantir que nenhum vídeo interferisse na codificação do outro.

Assim como apresentado no Capítulo 6, as perdas de eficiência de codificação foram avaliadas aplicando a métrica *Bjøntegaard Delta-Bitrate* (BD-BR), a Redução de Tempo (TR) foi calculada através da Equação (14) e a Redução de Tempo do TZS (TR_{TZS}) foi estimada utilizando a Equação (15).

O primeiro conjunto de experimentos utilizou uma versão do VTM apenas com os modelos de Aprendizado de Máquina desenvolvidos para o TZS. O segundo conjunto utilizou a versão do VTM com os modelos do TZS e da AME de forma conjunta.

8.1 Resultados da Solução para o TZS

Primeiro, foram realizados os experimentos aplicando somente o modelo de Aprendizado de Máquina do TZS com as Árvores de Decisão apresentadas na Tabela 15.

Os resultados, abrangendo todas as sequências de vídeo utilizadas, juntamente com os resultados médios para cada classe de vídeo e uma média geral, são apresentados na Tabela 19.

Tabela 19 – Resultados de redução de tempo e eficiência de codificação com a implementação do modelo do TZS.

Classe	Sequência	TR	TR_{TZS}	BD-BR
A1	Tango2	10,94%	40,37%	0,24%
	FoodMarket4	05,91%	22,33%	-0,09%
	Campfire	23,75%	69,80%	0,28%
	Média Classe A1	13,53%	44,17%	0,15%
A2	CatRobot	11,63%	52,02%	2,39%
	DaylightRoad2	11,96%	51,19%	0,15%
	ParkRunning3	17,78%	66,74%	0,18%
	Média Classe A2	13,79%	56,65%	0,91%
B	MarketPlace	28,36%	84,88%	1,58%
	RitualDance	28,85%	81,78%	1,20%
	Cactus	15,43%	61,57%	0,60%
	BasketballDrive	14,68%	54,84%	0,72%
	BQTerrace	10,60%	64,90%	0,12%
	Média Classe B	19,58%	69,59%	0,84%
C	BasketballDrill	20,18%	66,93%	-0,11%
	BQMall	16,59%	69,09%	0,70%
	PartyScene	20,58%	77,51%	0,78%
	RaceHorsesC	19,96%	70,15%	0,21%
	Média Classe C	19,33%	70,92%	0,39%
D	BasketballPass	13,66%	70,51%	0,22%
	BQSquare	09,92%	73,38%	0,41%
	BlowingBubbles	14,61%	71,42%	0,78%
	RaceHorses	20,63%	73,71%	0,37%
	Média Classe D	14,71%	72,26%	0,44%
F	BasketballDrillText	17,42%	65,79%	0,27%
	ArenaOfValor	17,76%	65,55%	0,93%
	SlideEditing	03,85%	54,75%	2,97%
	SlideShow	08,67%	46,81%	-0,42%
	Média Classe F	11,92%	58,22%	0,94%
	Média Geral	**15,48%**	**61,97%**	**0,61%**

Analisando a Tabela 19, observa-se que o VTM acelerado com as Árvores de Decisão para o TZS alcançou uma redução média de 15,48% no tempo total de codificação do VVC, além de reduzir o tempo de processamento do TZS em 61,97%. Ademais, verificou-se um leve decréscimo na eficiência de compressão, com um aumento médio

de 0,61% no BD-BR. A Classe C apresentou a segunda melhor redução no tempo de codificação, com um TR médio de 19,33%, e a segunda menor perda de eficiência de codificação, com um BD-BR médio de 0,39%. Assim, essa classe obteve o melhor *trade-off* entre TR e BD-BR. Por outro lado, a Classe F apresentou os piores resultados médios, com um TR de 11,92% e um BD-BR de 0,94%. Esse desempenho inferior ocorreu porque os quatro vídeos dessa classe são conteúdos de tela, enquanto as Árvores de Decisão foram treinadas exclusivamente em vídeos com cenas do mundo real.

Ademais, observa-se que algumas sequências de vídeo apresentaram resultados discrepantes em relação à média geral. Esse é o caso da sequência *FoodMarket4*, que apresentou uma redução modesta no tempo de codificação, de apenas 5,91%, e um ganho de eficiência de codificação em BD-BR, de 0,09%. Além disso, as sequências *SlideEditing* e *SlideShow*, da Classe F, também exibiram uma redução de tempo muito baixa. A primeira registrou uma perda de eficiência em BD-BR consideravelmente alta, atingindo 2,97%, enquanto a segunda apresentou um ganho em BD-BR de 0,42%. Esses resultados podem ser explicados pelo fato de ambas as sequências serem conteúdo de tela, diferindo das sequências utilizadas para treinar as Árvores de Decisão.

8.2 Resultados da Solução Completa

Em seguida, foram realizados os experimentos aplicando o modelo de Aprendizado de Máquina do TZS junto com o modelo da AME com as Árvores de Decisão apresentadas na Tabela 18. Este é o modelo de Aprendizado de Máquina completo deste trabalho.

Neste caso, uma nova métrica foi utilizada que é a Redução de Tempo da AME (TR_{AME}) para avaliar o quanto de tempo da AME em si foi reduzido. O cálculo da TR_{AME} está apresentado na Equação (16), em que T_{AME} é o tempo de execução da AME no VTM âncora, T_{AME_Mod} é o tempo de execução da AME no VTM modificado.

$$TR_{AME} = \frac{1}{4} \sum_{QP_i \epsilon \{22,27,32,37\}} \frac{T_{AME}(QP_i) - T_{AME\_Mod}(QP_i)}{T_{AME}(QP_i)} \tag{16}$$

Os resultados de todas as sequências de vídeo utilizadas com os resultados médios para cada classe de vídeo e uma média geral são apresentados na Tabela 20.

Tabela 20 – Resultados de redução de tempo e eficiência de codificação com a implementação dos modelos do TZS e da AME.

Classe	Sequência	TR	TR_{TZS}	TR_{AME}	BD-BR
A1	Tango2	17,87%	41,01%	53,38%	0,57%
	FoodMarket4	11,10%	22,75%	44,11%	-0,05%
	Campfire	26,39%	70,19%	59,09%	0,22%
	Média Classe A1	18,45%	44,65%	52,19%	0,25%
A2	CatRobot	20,18%	52,17%	59,59%	1,47%
	DaylightRoad2	21,53%	50,62%	61,38%	1,75%
	ParkRunning3	23,71%	66,73%	63,59%	0,49%
	Média Classe A2	21,81%	56,51%	61,50%	1,24%
B	MarketPlace	29,34%	85,16%	73,58%	1,90%
	RitualDance	30,36%	81,95%	73,65%	1,49%
	Cactus	22,22%	61,56%	64,20%	1,79%
	BasketballDrive	20,77%	54,85%	58,42%	0,96%
	BQTerrace	15,44%	65,53%	56,30%	0,13%
	Média Classe B	23,63%	69,81%	65,23%	1,25%
C	BasketballDrill	22,64%	65,26%	70,43%	-0,16%
	BQMall	22,64%	69,34%	71,49%	0,99%
	PartyScene	25,50%	77,88%	72,50%	1,45%
	RaceHorsesC	26,28%	70,36%	72,49%	0,80%
	Média Classe C	24,26%	70,72%	71,73%	0,77%
D	BasketballPass	21,75%	70,54%	75,14%	0,90%
	BQSquare	15,57%	73,72%	69,43%	0,94%
	BlowingBubbles	21,70%	71,90%	76,43%	0,57%
	RaceHorses	27,74%	74,24%	80,70%	1,67%
	Média Classe D	21,69%	72,60%	75,43%	1,02%
F	BasketballDrillText	22,84%	66,11%	69,62%	0,14%
	ArenaOfValor	23,88%	65,98%	62,34%	1,59%
	SlideEditing	05,45%	55,60%	39,20%	4,66%
	SlideShow	12,18%	46,79%	50,45%	-2,99%
	Média Classe F	16,09%	58,62%	55,40%	0,85%
	Média Geral	**20,99%**	**62,15%**	**63,58%**	**0,90%**

Com os resultados mostrados na Tabela 20, é possível observar que o VTM acelerado desenvolvido neste trabalho alcançou uma redução média de 20,99% no tempo total de codificação do VVC. Especificamente, reduziu o tempo de processamento do TZS em 62,15% e o tempo de processamento do AME em 63,58%. Além disso, foi observado um leve decréscimo na eficiência de compressão, com um aumento médio de 0,90% no BD-BR. Adicionalmente, o impacto das execuções das Árvores de Decisão no VTM foi mínimo, resultando em custo de apenas 0,30% no tempo total de codificação.

Analisando a Tabela 20, observa-se que os resultados da Classe A1 apresentaram o melhor *trade-off* entre TR médio, com 18,45%, e BD-BR médio, com 0,25%. Isso indica que a aceleração proposta funciona de forma especialmente eficiente para ví-

deos de alta resolução, o que é uma vantagem significativa, considerando a tendência contínua de aumento na resolução dos vídeos. Por outro lado, os resultados da Classe F são novamente os mais discrepantes. Do mesmo modo que na discussão anterior, essa discrepância ocorre porque todos os vídeos dessa classe são de conteúdo de tela, enquanto as Árvores de Decisão foram treinadas exclusivamente em vídeos com cenas do mundo real.

Comparando os resultados dos dois modelos apresentados nas Tabelas 19 e 20, observa-se que, embora a redução no tempo de codificação (TR) tenha aumentado, a perda na eficiência de codificação (BD-BR) também se elevou. Esse comportamento era esperado, uma vez que, para atingir maiores ganhos de tempo, as etapas da Predição Inter-Quadros são puladas mais vezes em relação ao processo original. Apesar disso, a solução completa manteve um *trade-off* muito satisfatório entre TR e BD-BR, com uma redução de tempo alta e uma perda de eficiência aceitável. Além disso, nota-se que os resultados de TR na solução completa são significativamente mais consistentes, com a maioria das sequências de vídeo apresentando reduções de tempo superiores a 20%.

Com a comparação entre estes resultados com os resultados obtidos com o experimento que apenas excluiu as três últimas etapas do TZS e excluiu a AME completa, apresentados na Tabela 11, os resultados atingidos pelas árvores de decisão foram muito bons. No experimento com a exclusão simples, foi obtida uma redução média de 29,39% no tempo total de codificação com um aumento de 9,84% no BD-BR. Então é possível observar que a solução proposta neste trabalho alcançou mais de 70% da redução máxima de tempo possível, com menos de 10% da perda de eficiência observada com a simples exclusão. Essa comparação mostra a eficiência do método proposto com o uso de Aprendizado de Máquina.

Apenas para registro, o tempo total necessário para codificar todas as sequências para a realização deste trabalho, incluindo a avaliação das ferramentas de Predição Inter-Quadros, extração de *features*, codificação com o VTM original e testes com o VTM acelerado, foi de aproximadamente 874 horas contínuas de processador.

8.3 Comparação com Trabalhos Relacionados

De forma a analisar melhor se os resultados obtidos foram satisfatórios, foi realizada uma comparação com alguns trabalhos relacionados descritos no Capítulo 5. Os resultados destes trabalhos estão apresentados na Tabela 5, onde foram incluídos apenas os trabalhos que reduzem o custo do TZS e da AME no VVC utilizando técnicas de Aprendizado de Máquina, para permitir uma comparação mais justa. Além disso, não foram considerados os trabalhos que utilizaram o VVenC ao invés do VTM, uma vez que as bases de comparação seriam completamente diferentes.

Uma versão inicial do trabalho de redução do tempo de execução do TZS foi publicada em **Viana et al. (2024a)**, mas o processo de construção dos modelos foi menos robusto, sem uma busca pelos melhores hiperparâmetros, sem balanceamento e resultando em *F1-Score* mais baixos para as Árvores de Decisão. Assim, os resultados muito competitivos que foram obtidos são função de um provável *overfitting* nos modelos e, por conta disso, um novo e robusto processo de modelagem foi desenvolvido e é apresentado neste texto.

Além disso, uma versão simplificada da redução do tempo de execução da AME também foi publicada em **Viana et al. (2024b)**. Assim como no caso do trabalho anterior para o TZS, o processo de construção dos modelos dessa versão foi menos robusto, sem busca pelos melhores hiperparâmetros, sem balanceamento dos dados e resultando em F1-Scores mais baixos para as Árvores de Decisão. Ambos os trabalhos estão apresentados no Apêndice B.

A comparação com trabalhos relacionados está apresentada na Tabela 21, onde são mostradas a versão do VTM, a ferramenta alvo, o método de ML utilizado, juntamente com os resultados de TR e BD-BR. Aqui foram incluídos os resultados de ambas as soluções desenvolvidas e apresentadas: solução para o TZS e a solução completa, com o TZS e a AME em conjunto.

Tabela 21 – Comparação dos resultados com trabalhos relacionados.

Trabalho	Versão VTM	Ferramenta Modificada	Método de ML	TR Médio	BD-BR Médio
Este Trabalho (Solução Completa)	**16.2**	**TZS e AME**	**DT**	**20,99%**	**0,90%**
Este Trabalho (Solução TZS)	**16.2**	**TZS**	**DT**	**15,48%**	**0,61%**
Duarte et al. (2022)	9.0	AME	RF	08,49%	0,18%
Huang et al. (2023)	10.0	AME	DT	10,33%	0,14%
Sagrilo et al. (2023)	16.2	AME	RF	02,98%	0,07%
Viana et al. (2024a)	16.2	TZS	DT	21,92%	0,45%
Viana et al. (2024b)	16.2	AME	DT	05,07%	0,23%

Analisando a Tabela 21 é possível observar que alguns trabalhos relacionados focaram em versões distintas do VTM, dificultando uma comparação mais justa, pois existem várias diferenças entre as versões do VTM, que impactam diretamente na frequência de uso das ferramentas e no tempo de processamento. Além disso, alguns trabalhos usaram Florestas Aleatórias (RF) em vez de Árvores de Decisão (DT). Mesmo assim, este trabalho alcançou resultados competitivos em comparação com todos os trabalhos relacionados, obtendo um bom *trade-off* entre o TR e o BD-BR.

Os resultados na Tabela 21 mostram que o trabalho desenvolvido com solução completa apresentou o segundo melhor resultado em ganho de tempo, com resultados próximos ao primeiro lugar, que, por sinal, é um trabalho anterior do autor desta

dissertação. Em relação aos demais trabalhos, os ganhos de tempo são entre 7,04 e 2,03 vezes maiores com a solução proposta neste trabalho. No geral, os trabalhos que usaram RF tiveram resultados de ganho de tempo inferiores aos trabalhos que usaram DTs. A hipótese para esse resultado é que o custo adicional de usar RFs acabou por impactar no ganho total de tempo.

Entre os trabalhos apresentados, o trabalho anterior desenvolvido pelo autor **Viana et al. (2024a)** foi o que alcançou os melhores resultados na relação entre TR e BD-BR. Ocorre que, como já discutido, esse trabalho foi desenvolvido com um processo de modelagem menos rigoroso e, assim, os resultados dos modelos apresentados nesta tese são mais confiáveis do ponto de vista de sua generalização para aplicações no mundo real.

Em relação aos resultados de perda de eficiência de codificação dos trabalhos apresentados na Tabela 21, é possível perceber que o trabalho desenvolvido nesta dissertação, mesmo apresentando um impacto inferior a 1%, foi a solução com maior impacto na eficiência de codificação. Esse resultado é esperado na medida em que a solução proposta é mais agressiva para atingir ganhos significativos no tempo total de processamento.

Por fim, é importante destacar que este trabalho é o único que atua na redução de tempo de múltiplas ferramentas de codificação, demonstrando que o uso de Aprendizado de Máquina pode melhorar de maneira eficaz diversos aspectos das implementações do VVC.

9 CONCLUSÃO

O padrão VVC é capaz de alcançar as maiores taxas de compressão entre os codificadores atuais, mas essa vantagem vem acompanhada de demandas computacionais substanciais. No modelo de teste do VVC, os algoritmos de *Test Zone Search* e Estimação de Movimento *Affine* estão entre os mais computacionalmente custosos, representando juntos mais de 35% do tempo total de codificação.

Este trabalho apresentou soluções de Aprendizado de Máquina para reduzir o esforço computacional das ferramentas TZS e AME no VVC. A solução utiliza modelos de Árvores de Decisão para pular seletivamente três das quatro etapas do TZS e todo o processo da AME. Para cada ferramenta, foram desenvolvidas 12 Árvores de Decisão, uma para cada tamanho de bloco suportado pelas ferramentas. Assim, para cada tamanho de bloco processado, uma Árvore de Decisão específica toma a decisão se o processo alvo deve ser pulado ou executado. Essa estratégia foi avaliada em duas etapas. Inicialmente, foram treinados e avaliados os modelos para a aceleração do TZS no software de referência VTM, pulando seletivamente as três últimas etapas do TZS. Em seguida, usando a implementação com a aceleração do TZS, foram desenvolvidos os modelos de aceleração na AME, pulando seletivamente todo o seu processo.

O software VTM acelerado com os dois conjuntos de Árvores de Decisão alcançou uma redução média de 20,99% no tempo total de codificação do VVC. Especificamente, reduziu o tempo médio de processamento do TZS em 62,15% e o tempo médio de processamento da AME em 63,58%, com uma perda média de eficiência de codificação de 0,90% em BD-BR. Esta aceleração do VVC alcançou uma redução no tempo de codificação expressiva, mantendo uma baixa perda de eficiência de codificação.

A solução proposta alcançou resultados competitivos quando comparada com outros trabalhos da literatura, conseguindo um bom *trade-off* entre redução de tempo de processamento e perda de eficiência de codificação. Não foi encontrado nenhum trabalho na literatura que acelere duas ferramentas da Predição Inter-Quadros do VVC ao mesmo tempo. Portanto, este trabalho é o primeiro que o faz e os resultados atingidos demonstram que essa é uma estratégia interessante para conseguir bons resultados.

Como trabalho futuro, pretende-se explorar a aceleração dos diferentes tipos de particionamento de bloco no VVC por meio de Aprendizado de Máquina, considerando que estudos anteriores já demonstraram bons resultados nessa abordagem. Essas acelerações poderão ser integradas à solução proposta nesta dissertação.

Além disso, outro trabalho futuro será a aceleração de outras ferramentas críticas do VVC, como as Transformadas e a Predição Intra-Quadros. Outra linha de pesquisa será o trabalho com vídeos 360º, uma área ainda pouco explorada nas soluções com o VVC, mas que apresenta um enorme potencial de aplicação no futuro.

REFERÊNCIAS

AGOSTINI, L. V. **Desenvolvimento de Arquiteturas de Alto Desempenho Dedicadas à Compressão de Vídeo Segundo o Padrão H. 264/AVC**. 2007. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio Grande so Sul.

AMESTOY, T.; MERCAT, A.; HAMIDOUCHE, W.; MENARD, D.; BERGERON, C. Tunable VVC Frame Partitioning Based on Lightweight Machine Learning. **IEEE Transactions on Image Processing**, [S.l.], v.29, p.1313–1328, 2019.

BENJAK, M.; MEUEL, H.; LAUDE, T.; OSTERMANN, J. Enhanced Machine Learning-based Inter Coding for VVC. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE IN INFORMATION AND COMMUNICATION (ICAIIC), 2021., 2021. **Anais...** [S.l.: s.n.], 2021. p.021–025.

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. **The Journal of Machine Learning Research**, [S.l.], v.13, p.281–305, 2012.

BHASKARAN, V.; KONSTANTINIDES, K. **Image and video compression standards: algorithms and architectures**. 1.ed. [S.l.]: Springer Science & Business Media, 1997.

BJONTEGAARD, G. **Calculation of average PSNR differences between RD-curves**. Technical Report VCEG-M33, ITU-TSG16/Q6, Austin, Texas, USA.

BOSSEN, F.; BOYCE, J.; LI, X.; SEREGIN, V.; SUHRING, K. **VTM common test conditions and software reference configurations for SDR video**. JVET-T2010. Disponível em: <$https://jvet-experts.org/doc_end_user/current_document.php?id=10545$>. Acesso em: 28/04/2023.

BOUKHATEM, C.; YOUSSEF, H. Y.; NASSIF, A. B. Heart Disease Prediction Using Machine Learning. In: ADVANCES IN SCIENCE AND ENGINEERING TECHNOLOGY INTERNATIONAL CONFERENCES (ASET), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.1–6.

BOYCE, J.; SUEHRING, K.; LI, X.; SEREGIN, V. **JVET common test conditions and software reference configurations**. Joint Video Experts Team (JVET)of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11.

BROSS, B.; CHEN, J.; LIU, S.; WANG, Y.-K. **Versatile Video Coding Editorial Refinements on Draft 10**. JVET-T2001. Disponível em: <$https://jvet-experts.org/doc_end_user/current_document.php?id=10540$>. Acesso em: 28/04/2023.

BROSS, B.; CHEN, J.; OHM, J.-R.; SULLIVAN, G. J.; WANG, Y.-K. Developments in International Video Coding Standardization After AVC, With an Overview of Versatile Video Coding (VVC). **Proceedings of the IEEE**, [S.l.], v.109, n.9, p.1463–1493, 2021.

BROSS, B.; WANG, Y.-K.; YE, Y.; LIU, S.; CHEN, J.; SULLIVAN, G. J.; OHM, J.-R. Overview of the Versatile Video Coding (VVC) Standard and its Applications. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.31, n.10, p.3736–3764, 2021.

BROWNE, A.; YE, Y.; KIM, S. H. Algorithm description for Versatile Video Coding and Test Model 17 (VTM 17). **Jt. Video Expert. Team ITU-T SG 16 WP 3 ISO/IEC JTC 1/SC 29, 26th Meet. by teleconference**, [S.l.], April 2022.

BURKOV, A. **The Hundred-Page Machine Learning Book**. 1.ed. [S.l.]: Andriy Burkov, 2019.

CHEN, B.; WANG, Z.; LI, B.; WANG, S.; YE, Y. Compact Temporal Trajectory Representation for Talking Face Video Compression. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], p.1–14, 2023.

CHEN, J.; YE, Y.; KIM, S. H. Algorithm description for Versatile Video Coding and Test Model 11 (VTM 11). **Jt. Video Expert. Team ITU-T SG 16 WP 3 ISO/IEC JTC 1/SC 29/WG 11, 20th Meet. by teleconference**, [S.l.], October 2020.

CHEN, M.-J.; LEE, C.-A.; TSAI, Y.-H.; YANG, C.-M.; YEH, C.-H.; KAU, L.-J.; CHANG, C.-Y. Efficient Partition Decision Based on Visual Perception and Machine Learning for H.266/Versatile Video Coding. **IEEE Access**, [S.l.], v.10, p.42141–42150, 2022.

CHIEN, W.-J.; ZHANG, L.; WINKEN, M.; LI, X.; LIAO, R.-L.; GAO, H.; HSU, C.-W.; LIU, H.; CHEN, C.-C. Motion Vector Coding and Block Merging in the Versatile Video Coding Standard. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.31, n.10, p.3848–3861, 2021.

CHOI, K. A Study on Fast and Low-Complexity Algorithms for Versatile Video Coding. **Sensors**, [S.l.], v.22, n.22, 2022.

CISCO. **Cisco Annual Internet Report (2018–2023)**. Disponível em: <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html>. Acesso em: 08/02/2024.

COADY, J.; O'RIORDAN, A.; DOOLY, G.; NEWE, T.; TOAL, D. An Overview of Popular Digital Image Processing Filtering Operations. In: INTERNATIONAL CONFERENCE ON SENSING TECHNOLOGY (ICST), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.1–5.

CORRÊA, G. R. **Computational Complexity Reduction and Scaling for High Efficiency Video Encoders**. 2014. Tese (Doutorado em Ciência da Computação) — Universidade de Coimbra.

DAEDE, T.; NORKIN, A.; BRAILOVSKIY, I. **Video Codec Testing and Quality Measurement**. Accessed: 2022-02-22, `https://datatracker.ietf.org/doc/html/draft-ietf-netvc-testing-07`.

DOAN, N.; KIM, T. S.; RHEE, C. E.; LEE, H.-J. A hardware-oriented concurrent TZ search algorithm for High-Efficiency Video Coding. **EURASIP Journal on Advances in Signal Processing**, [S.l.], v.2017, p.78, November 2017.

DUARTE, A.; GONÇALVES, P.; AGOSTINI, L.; ZATT, B.; CORREA, G.; PORTO, M.; PALOMINO, D. Fast Affine Motion Estimation for VVC using Machine-Learning-Based Early Search Termination. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.1–5.

DUARTE, A. I. R. **Redução de complexidade do processo de decisão de modo da predição intra-quadro do codificador de vídeo VVC utilizando aprendizado de máquina**. 2021. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Pelotas.

FERREIRA, R.; AGOSTINI, L.; DINIZ, C. M.; ZATT, B. Evaluation of Imprecise Subtractors into Test Zone Search for VVC Encoding. In: SBC/SBMICRO/IEEE/ACM SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN (SBCCI), 2023., 2023. **Anais...** [S.l.: s.n.], 2023. p.1–6.

FERREIRA, R.; SANTOS, L.; AGOSTINI, L.; DINIZ, C. M.; ZATT, B. VVC Interpicture Prediction Using SAD with Imprecise Subtractors: A Quantitative Analysis. In: IEEE INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS AND SYSTEMS (ICECS), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.1–4.

GAO, H.; ESENLIK, S.; ALSHINA, E.; STEINBACH, E. Geometric Partitioning Mode in Versatile Video Coding: Algorithm Review and Analysis. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.31, n.9, p.3603–3617, 2021.

GONÇALVES, P.; CORREA, G.; AGOSTINI, L.; PORTO, M. Learning-based bypass zone search algorithm for fast motion estimation. **Multimedia Tools and Applications**, [S.l.], p.3535–3560, 2022.

GONÇALVES, P.; CORREA, G.; PORTO, M.; ZATT, B.; AGOSTINI, L. Multiple early-termination scheme for TZ search algorithm based on data mining and decision trees. In: IEEE 19TH INTERNATIONAL WORKSHOP ON MULTIMEDIA SIGNAL PROCESSING (MMSP), 2017., 2017. **Anais...** [S.l.: s.n.], 2017. p.1–6.

GONÇALVES, P. H. R. **Um esquema rápido baseado em aprendizado de máquina para a predição interquadros do codificador de vídeo VVC**. 2021. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Pelotas.

GONZALEZ, R. C.; WOODS, R. E. **Processamento Digital de Imagens**. 3.ed. [S.l.]: Pearson Universidades, 2010.

GUAN, X.; SUN, X. VVC Fast ME Algorithm Based on Spatial Texture Features and Time Correlation. In: INTERNATIONAL CONFERENCE ON DIGITAL SOCIETY AND INTELLIGENT SYSTEMS (DSINS), 2021., 2021. **Anais...** [S.l.: s.n.], 2021. p.371–377.

GUPTA, V.; MISHRA, V. K.; SINGHAL, P.; KUMAR, A. An Overview of Supervised Machine Learning Algorithm. In: INTERNATIONAL CONFERENCE ON SYSTEM MODELING & ADVANCEMENT IN RESEARCH TRENDS (SMART), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.87–92.

HAMIDOUCHE, W.; BIATEK, T.; ABDOLI, M.; FRANçOIS, E.; PESCADOR, F.; RADOSAVLJEVIć, M.; MENARD, D.; RAULET, M. Versatile Video Coding Standard: A Review From Coding Tools to Consumers Deployment. **IEEE Consumer Electronics Magazine**, [S.l.], v.11, n.5, p.10–24, 2022.

HUANG, X.; ZHOU, F.; NIU, W.; LI, T.; LU, Y.; ZHOU, Y.; YIN, H.; YAN, C. Multi-stage affine motion estimation fast algorithm for versatile video coding using decision tree. **Journal of Visual Communication and Image Representation**, [S.l.], v.96, p.103910, 2023.

JVET - Joint Video Experts Team. **VTM - VVC Test Model**. Disponível em: <$https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM$>. Acesso em: 04/02/2025.

JVET - Joint Video Experts Team. **VVenC - Fraunhofer Versatile Video Encoder**. Disponível em: <$https://github.com/fraunhoferhhi/vvenc/wiki$>. Acesso em: 04/02/2025.

KULUPANA, G.; KUMAR M, V. P.; BLASI, S. Fast Versatile Video Coding using Specialised Decision Trees. In: PICTURE CODING SYMPOSIUM (PCS), 2021., 2021. **Anais...** [S.l.: s.n.], 2021. p.1–5.

LI, J.; ZHANG, S.; YANG, F. Random Forest Accelerated CU Partition for Inter Prediction in H.266/VVC. In: IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA AND EXPO (ICME), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.01–06.

LI, L.; WU, K.; WEI, H.; LIU, J.; FANG, Y.; ZHENG, H. Deep Motion Vector Prediction for Versatile Video Coding. In: INTERNATIONAL CONFERENCE ON COMPUTER AND COMMUNICATIONS (ICCC), 2021., 2021. **Anais...** [S.l.: s.n.], 2021. p.874–878.

LIN, J.; LIN, H.; ZHANG, Z.; XU, Y. Efficient inter partitioning of versatile video coding based on supervised contrastive learning. **Knowledge-Based Systems**, [S.l.], v.296, p.111902, 2024.

LINDINO, M.; ZATT, B.; GRELLERT, M.; CORREA, G. Low-Complexity Multi-Type Tree Partitioning for Versatile Video Coding Based on Machine Learning. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.3616–3620.

LIU, H.; ZHANG, L.; ZHANG, K.; XU, J.; WANG, Y.; LUO, J.; HE, Y. Adaptive Motion Vector Resolution for Affine-Inter Mode Coding. In: PICTURE CODING SYMPOSIUM (PCS), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.1–4.

LOOSE, M.; VIANA, R.; SAGRILO, F.; SANCHEZ, G.; CORRÊA, G.; AGOSTINI, L. A Hardware-Friendly and Configurable Heuristic Targeting VVC Inter-Frame Prediction. In: IEEE INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS AND SYSTEMS (ICECS), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.1–4.

MAASS, D.; MUÑOZ, M.; PERLEBERG, M.; AGOSTINI, L.; PORTO, M. High-Throughput Hardware Design for Linear Equation System Solving of VVC Affine Prediction. **Journal of Integrated Circuits and Systems**, [S.l.], v.18, n.3, p.1–11, 2023.

MAASS, D.; MUNOZ, M.; PERLEBERG, M.; AGOSTINI, L.; PORTO, M. A Real-Time UHD 4K Hardware for VVC Affine Linear Equation System Solving. In: SBC/SBMICRO/IEEE SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN (SBCCI), 2024., 2024. **Anais...** [S.l.: s.n.], 2024. p.1–5.

MAIMON, O.; ROKACH, L. **Data Mining and Knowledge Discovery Handbook**. 1.ed. [S.l.]: Springer US, 2005.

MARTINS, A.; PENNY, W.; WEBER, M.; PALOMINO, D.; MATTOS, J.; PORTO, M.; AGOSTINI, L.; ZATT, B. Cache Memory Energy Efficiency Exploration for the HEVC

Motion Estimation. In: VII BRAZILIAN SYMPOSIUM ON COMPUTING SYSTEMS ENGINEERING (SBESC), 2017., 2017. **Anais...** [S.l.: s.n.], 2017. p.31–38.

MERCAT, A.; MäKINEN, A.; SAINIO, J.; LEMMETTI, A.; VIITANEN, M.; VANNE, J. Comparative Rate-Distortion-Complexity Analysis of VVC and HEVC Video Codecs. **IEEE Access**, [S.l.], v.9, p.67813–67828, 2021.

MERCAT, A.; VIITANEN, M.; VANNE, J. UVG dataset. In: ACM MULTIMEDIA SYSTEMS CONFERENCE, 11., 2020. **Proceedings...** ACM, 2020.

MUÑOZ, M. M.; MAASS, D.; PERLEBERG, M.; AGOSTINI, L.; PORTO, M. Efficient Hardware Design for the VVC Affine Motion Compensation Exploiting Multiple Constant Multiplication. In: IEEE COMPUTER SOCIETY ANNUAL SYMPOSIUM ON VLSI (ISVLSI), 2023., 2023. **Anais...** [S.l.: s.n.], 2023. p.1–6.

PALAU, R.; SILVEIRA, B.; DOMANSKI, R.; LOOSE, M.; CERVEIRA, A.; SAMPAIO, F.; PALOMINO, D.; PORTO, M.; CORRÊA, G.; AGOSTINI, L. Modern Video Coding: Methods, Challenges and Systems. **Journal of Integrated Circuits and Systems**, [S.l.], v.16, n.2, p.1–12, 2021.

PAN, Z.; ZHANG, P.; PENG, B.; LING, N.; LEI, J. A CNN-Based Fast Inter Coding Method for VVC. **IEEE Signal Processing Letters**, [S.l.], v.28, p.1260–1264, 2021.

PAPAGEORGIOU, E.; STYLIOS, C.; GROUMPOS, P. A Combined Fuzzy Cognitive Map and Decision Trees Model for Medical Decision Making. In: INTERNATIONAL CONFERENCE OF THE IEEE ENGINEERING IN MEDICINE AND BIOLOGY SOCIETY, 2006., 2006. **Anais...** [S.l.: s.n.], 2006. p.6117–6120.

PARK, J.; KIM, B.; LEE, J.; JEON, B. Machine Learning-Based Early Skip Decision for Intra Subpartition Prediction in VVC. **IEEE Access**, [S.l.], v.10, p.111052–111065, 2022.

PARK, S.-H.; KANG, J.-W. Fast Affine Motion Estimation for Versatile Video Coding (VVC) Encoding. **IEEE Access**, [S.l.], v.7, p.158075–158084, 2019.

PEJMAN, H.; COULOMBE, S.; VAZQUEZ, C.; JAMALI, M.; VAKILI, A. An Adjustable Fast Decision Method for Affine Motion Estimation in VVC. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2023., 2023. **Anais...** [S.l.: s.n.], 2023. p.2695–2699.

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. Systematic Mapping Studies in Software Engineering. In: INTERNATIONAL CONFERENCE ON EVALUATION AND ASSESSMENT IN SOFTWARE ENGINEERING, 12., 2008, Swindon, UK. **Proceedings...** BCS Learning & Development Ltd., 2008. p.68–77. (EASE'08).

PORTO, M. S. **Desenvolvimento Algorítmico e Arquitetural para a Estimação de Movimento na Compressão de Vídeo de Alta Definição**. 2012. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio Grande so Sul.

PRASAD, N.; KUMAR, P.; MM, N. An Approach to Prediction of Precipitation Using Gini Index in SLIQ Decision Tree. In: INTERNATIONAL CONFERENCE ON INTELLIGENT SYSTEMS, MODELLING AND SIMULATION, 2013., 2013. **Anais...** [S.l.: s.n.], 2013. p.56–60.

RICHARDSON, I. **Video Codec Design: Developing Image and Video Compression Systems**. 1.ed. [S.l.]: Wiley, 2002.

SAGRILO, F.; LOOSE, M.; VIANA, R.; SANCHEZ, G.; CORRÊA, G.; AGOSTINI, L. Learning-Based Fast VVC Affine Motion Estimation. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS), 2023., 2023. **Anais...** [S.l.: s.n.], 2023. p.1–5.

SALDANHA, M.; CORRÊA, M.; CORRÊA, G.; PALOMINO, D.; PORTO, M.; ZATT, B.; AGOSTINI, L. 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., 2020. **Anais...** [S.l.: s.n.], 2020. p.1–4.

SANT'ANNA, G. B.; HENRIQUE CANCELLIER, L.; SEIDEL, I.; GRELLERT, M.; GüNTZEL, J. L. Relying on a Rate Constraint to Reduce Motion Estimation Complexity. In: ICASSP 2021 - 2021 IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING (ICASSP), 2021. **Anais...** [S.l.: s.n.], 2021. p.1560–1564.

SHEN, L.; YANG, H.; WANG, S. Effective QTMT Partition Decision Algorithm for VVC Intercoding. In: IEEE 24TH INTERNATIONAL WORKSHOP ON MULTIMEDIA SIGNAL PROCESSING (MMSP), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.1–6.

SHENG, Q.; CHEN, H.; LAI, C.; HUANG, X.; LIU, Y.; HUANG, X.; YIN, H. Fast Linear Equation Solving Algorithm and its Pipelined Hardware Architecture Design for VVC Affine Motion Estimation. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], p.1–1, 2024.

SINGH, K.; AHAMED, S. R. Computationally efficient motion estimation algorithm for HEVC. **Journal of Signal Processing Systems**, [S.l.], v.90, p.1713–1727, 2018.

SIQUEIRA, I.; CORREA, G.; GRELLERT, M. Rate-Distortion and Complexity Comparison of HEVC and VVC Video Encoders. In: IEEE 11TH LATIN AMERICAN SYMPO-

SIUM ON CIRCUITS AND SYSTEMS (LASCAS), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.1–4.

SIQUEIRA, I.; CORREA, G.; GRELLERT, M. Complexity and Coding Efficiency Assessment of the Versatile Video Coding Standard. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS), 2021., 2021. **Anais...** [S.l.: s.n.], 2021. p.1–5.

SOBEL, I. An Isotropic 3x3 Image Gradient Operator. **Presentation at Stanford A.I. Project 1968**, [S.l.], February 2014.

SOMVANSHI, M.; CHAVAN, P.; TAMBADE, S.; SHINDE, S. V. A review of machine learning techniques using decision tree and support vector machine. In: INTERNATIONAL CONFERENCE ON COMPUTING COMMUNICATION CONTROL AND AUTOMATION (ICCUBEA), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.1–7.

STORCH, I. C. **Exploração das distorções da projeção ERP para redução de complexidade da codificação de vídeos omnidirecionais**. 2020. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Pelotas.

STORCH, I.; PALOMINO, D.; BAMPI, S. GPU-Acceleration of Affine Prediction in the Versatile Video Coding. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.429–433.

SUEHRING, K. **VTM-16.2**. Disponível em: <$https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/releases/VTM-16.2$>. Acesso em: 28/04/2023.

SUN, Y.-C.; LOU, J.; CHAO, Y.-H.; WANG, H.; SEREGIN, V.; KARCZEWICZ, M. Analysis of Palette Mode on Versatile Video Coding. In: IEEE CONFERENCE ON MULTIMEDIA INFORMATION PROCESSING AND RETRIEVAL (MIPR), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.455–458.

SUTHAHARAN, S. **Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning**. New York, NY: Springer, 2016.

TAABANE, I.; MENARD, D.; MANSOURI, A.; SAHRAOUI, S.; AHAITOUF, A. Low Complexity Learning-Based QTMTT Partitioning Scheme for Inter Coding in VVC Encoder. **IEEE Access**, [S.l.], v.12, p.141088–141103, 2024.

TISSIER, A.; HAMIDOUCHE, W.; VANNE, J.; MENARD, D. Machine Learning Based Efficient Qt-Mtt Partitioning for VVC Inter Coding. In: IEEE INTERNATIONAL CON-

FERENCE ON IMAGE PROCESSING (ICIP), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.1401–1405.

VAYALIL, N. C.; PAUL, M.; KONG, Y. A novel angle-restricted test zone search algorithm for performance improvement of HEVC. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2017., 2017. **Anais...** [S.l.: s.n.], 2017. p.6–10.

VIANA, R.; LOOSE, M.; CONCEIÇÃO, R.; PORTO, M.; CORRÊA, G.; AGOSTINI, L. Fast VVC Test Zone Search and Affine Motion Estimation Using Machine Learning. In: IEEE 16TH LATIN AMERICA SYMPOSIUM ON CIRCUITS AND SYSTEMS (LASCAS), 2025., 2025. **Anais...** [S.l.: s.n.], 2025. p.1–5.

VIANA, R.; LOOSE, M.; FERREIRA, R.; PORTO, M.; CORRÊA, G.; AGOSTINI, L. A Hardware-Friendly Acceleration of VVC Affine Motion Estimation Using Decision Trees. In: SBC/SBMICRO/IEEE SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN (SBCCI), 2024., 2024. **Anais...** [S.l.: s.n.], 2024. p.1–5.

VIANA, R.; SAGRILO, F.; FERREIRA, R.; LOOSE, M.; PORTO, M.; CORRÊA, G.; AGOSTINI, L. A Hardware-Friendly Fast VVC Test Zone Search Algorithm Using Machine Learning. In: IEEE 15TH LATIN AMERICA SYMPOSIUM ON CIRCUITS AND SYSTEMS (LASCAS), 2024., 2024. **Anais...** [S.l.: s.n.], 2024. p.1–5.

WEI, X.; ZENG, H.; FANG, Y.; LIN, L.; CHEN, W.; XU, Y. Multi-feature fusion for efficient inter prediction in versatile video coding. **Journal of Real-Time Image Processing**, [S.l.], v.21, p.1–14, 2024.

XIE, K.; ZHOU, J.; ZHANG, S.; YANG, F. Fast Inter Prediction Mode Decision Method Based On Random Forest For H.266/VVC. In: IEEE INTERNATIONAL CONFERENCE ON VISUAL COMMUNICATIONS AND IMAGE PROCESSING (VCIP), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.1–5.

XU, J.; HE, S. Optimization of Inter-Frame Coding Algorithm Based on Random Forest. In: ASIAN CONFERENCE ON ARTIFICIAL INTELLIGENCE TECHNOLOGY (ACAIT), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.1–4.

YU, X.; YANG, F. The Optimization of TZSearch for VVC Motion Estimation. In: INTERNATIONAL CONFERENCE ON UBIQUITOUS COMMUNICATION (UCOM), 2023., 2023. **Anais...** [S.l.: s.n.], 2023. p.156–159.

Apêndices

APÊNDICE A – Arquivos Utilizados Neste Trabalho

A.1 Arquivos VTM

- VTM Modificado para Extrair Tempos da Predição Inter-Quadros
  - `https://github.com/rgsviana/VTM-16.2-Optimization-00-Inter-Frame-Times`
- VTM Modificado para Extrair *Features* do TZS
  - `https://github.com/rgsviana/VTM-16.2-Optimization-01-Features-Extraction-TZS`
- VTM Modificado com as DTs do TZS Implementadas
  - `https://github.com/rgsviana/VTM-16.2-Optimization-02-DTs-Implementation-TZS`
- VTM Modificado para Extrair *Features* da AME
  - `https://github.com/rgsviana/VTM-16.2-Optimization-03-Features-Extraction-AME`
- VTM Modificado com as DTs da AME Implementadas
  - `https://github.com/rgsviana/VTM-16.2-Optimization-04-DTs-Implementation-AME`
- VTM Modificado Para Sempre Pular o TZS e a AME
  - `https://github.com/rgsviana/VTM-16.2-Optimization-05-TZS-AME-Jumps`

A.2 Arquivos em Python

- *Scripts* para Organizar *Datasets* e Gerar Árvores de Decisão do TZS
  - `https://github.com/rgsviana/VTM-DTs-TZS-Creation`
- *Scripts* para Organizar *Datasets* e Gerar Árvores de Decisão da AME
  - `https://github.com/rgsviana/VTM-DTs-AME-Creation`
- *Script* para Buscar os Melhores Hiperparâmetros com *Random Search*
  - `https://github.com/rgsviana/Hyperparameters-Random-Search`

APÊNDICE B – Trabalhos Publicados Durante o Período de Mestrado

B.1 A Hardware-Friendly Fast VVC Test Zone Search Algorithm Using Machine Learning

- **Autores:** Ramiro Viana, Fernando Sagrilo, Rafael Ferreira, Marta Loose, Marcelo Porto, Guilherme Corrêa e Luciano Agostini.
- **Conferência:** 15th IEEE Latin American Symposium on Circuits and Systems (LASCAS 2024).
- **Abstract:** The Versatile Video Coding (VVC) currently stands as the state-of-the-art in video coding efficiency, but with the cost of substantial computational effort and, consequently, high energy consumption. This elevated energy usage poses a critical challenge, particularly for applications designed for battery-powered devices. A key component of the VVC is the Test Zone Search (TZS) algorithm, known for its demanding computational requirements. This paper addresses this issue by introducing a machine learning based solution for the TZS algorithm. Notably, this solution is hardware-friendly, leveraging Decision Trees that are straightforward to implement in hardware. The proposed solution achieved an average 86.98% reduction in TZS encoding time with a mere 0.45% impact on BD-BR.

B.2 A Hardware-Friendly Acceleration of VVC Affine Motion Estimation Using Decision Trees

- **Autores:** Ramiro Viana, Marta Loose, Rafael Ferreira, Marcelo Porto, Guilherme Corrêa e Luciano Agostini.
- **Conferência:** 37th Symposium on Integrated Circuits and systems Design (SBCCI 2024).
- **Abstract:** Digital videos have become ubiquitous across various devices, presenting challenges in transmitting and storing due to their substantial data volume. Video encoders play a vital role in addressing these challenges by improving data compression. The Versatile Video Coding (VVC) encoder, designed to

enhance encoding processes compared to its predecessors, introduces a novel tool for Inter-Frame Prediction called Affine Motion Estimation (AME). This paper focuses on the Affine tool inside the Inter-Frame Prediction step, with the aim of developing an optimized, hardware-friendly algorithm. The goal is to reduce computational effort while maintaining the coding efficiency of the VVC encoder. To achieve this, Machine Learning models based on Decision Trees, known for their simplicity in implementation, are employed. The proposed solution successfully achieves an average time reduction of 5.07% in total encoding time, with a minimal coding efficiency loss of 0.23%.

B.3 Fast VVC Test Zone Search and Affine Motion Estimation Using Machine Learning

- **Autores:** Ramiro Viana, Marta Loose, Ruhan Conceição, Marcelo Porto, Guilherme Corrêa e Luciano Agostini.
- **Conferência:** 16th IEEE Latin American Symposium on Circuits and Systems (LASCAS 2025).
- **Abstract:** As the demand for video transmission surges on remote work, education, and streaming services, the need for continuous advancements in video encoding technologies becomes increasingly evident. Adapting to the evolving requirements of efficient video delivery and consumption necessitates ongoing development and enhancement in video encoding standards, with Versatile Video Coding (VVC) emerging as a notable example. This paper provides an overview of key algorithms within Inter-Frame prediction of VVC, mainly focusing on the Test Zone Search (TZS) and the Affine Motion Estimation (AME), two of the most computationally intensive tools inside the VVC. Furthermore, this paper introduces a fast TZS and AME approach using Machine Learning, specifically employing Decision Trees. The proposed approach achieved an average reduction of over 20% in total VVC encoding time while maintaining less than a 1% impact on coding efficiency in BD-BR.

B.4 Learning-Based Fast VVC Affine Motion Estimation

- **Autores:** Fernando Sagrilo, Marta Loose, Ramiro Viana, Gustavo Sanchez, Guilherme Corrêa e Luciano Agostini.
- **Conferência:** 58th IEEE International Symposium on Circuits and Systems (ISCAS 2023).

- **Abstract:** This paper presents a fast Affine Motion Estimation (AME) of Versatile Video Coding (VVC) Standard, based on Machine Learning and using Random Forest (RF) classification method. This encoding approach develops an RF model for each block size. The models were trained with information extracted during the VVC encoding process of the current, parent, and neighboring Coding Units (CU). Each model is applied to predict whether the Affine Motion Estimation (AME) will be skipped or not for that CU size. The proposed solution achieves a reduction of 20% on average in AME encoding time, with an insignificant impact of 0.07% on BD-BR.

B.5 High-Throughput Hardware Design for the AV1 Decoder Switchable Loop Restoration Filters

- **Autores:** Roberta Palau, Wagner Penny, Ramiro Viana, Jones Goebel, Marcelo Porto, Guilherme Corrêa e Luciano Agostini.

- **Periódico:** Journal of Integrated Circuits and Systems (JICS) Vol. 18 No. 1 (2023).

- **Abstract:** This paper presents a high-throughput hardware design for the Switchable Loop Restoration Filter (SLRF) of the AOM Video 1 (AV1) video format. This hardware includes the two filters defined at the AV1 SLRF: the Separable Symmetric Normalized Wiener Filter (SSNWF) and the Dual Self-Guided Filter (DSGF). The SLRF is the last step in the AV1 loop restoration filters, and it is used to attenuate blurring artifacts, improving the subjective video quality and the coding efficiency. The designed hardware targeted the AV1 decoder and is able to process up to 4K Ultra-High Definition (4K UHD) videos (with 3840x2160 píxeis) at 60 frames per second (fps) in real-time. In order to cover different scenarios, two other target throughputs were also evaluated: 4K UHD at 30fps and Full HD (FHD) (with 1920x1080 píxeis) at 30fps. The architectures were synthesized for standard cells using the 40 nm TSMC library. The SSNWF and DSGF architectures used 37.38 Kgates and 177.58 kgates in all evaluated scenarios. Depending on the evaluated scenario, the SSNWF power dissipation varied from 8.25 mW to 26.95 mW and the DSGF power varied from 57.19 mW to 115.02 mW. This is the first paper in the literature presenting a hardware design for the AV1 SLRF with its two filters.