UNIVERSIDADE FEDERAL DO PAMPA

Fernando Funghetto Sagrilo

Redução de Complexidade no Affine Motion Estimation do Padrão VVC

Alegrete
2022

Fernando Funghetto Sagrilo

Redução de Complexidade no Affine Motion Estimation do Padrão VVC

Dissertação apresentada ao Programa de Pós-graduação Stricto Sensu em Engenharia Elétrica da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Gustavo Freitas Sanchez

Alegrete
2022

Ficha catalográfica elaborada automaticamente com os dados fornecidos
pelo(a) autor(a) através do Módulo de Biblioteca do
Sistema GURI (Gestão Unificada de Recursos Institucionais) .

S129r Sagrilo, Fernando Funghetto
Redução de complexidade no affine motion estimation do padrão VVC / Fernando Funghetto Sagrilo.
79 p.

Dissertação(Mestrado)-- Universidade Federal do Pampa, MESTRADO EM ENGENHARIA ELÉTRICA, 2022.
"Orientação: Gustavo Freitas Sanchez".

1. VVC. 2. Inter-quadros. 3. Affine. 4. Machine Learning. 5. Random Forest. I. Título.

FERNANDO FUNGHETTO SAGRILO

REDUÇÃO DE COMPLEXIDADE NO AFFINE MOTION ESTIMATION DO PADRÃO VVC

Dissertação/Tese apresentada ao Programa de Pós–graduação em Engenharia Elétrica, da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Mestre em Engenharia Elétrica.

Dissertação defendida e aprovada em: 13/12/2022

Banca examinadora:

_____

Prof. Dr. Gustavo Freitas Sanchez

Orientador

(IFSertãoPE)

_____

Prof. Dr. Fábio Rossi Diniz

(IFFar)

_____

Prof. Dr. Daniel Palomino

(UFPEL)

Prof. Dr. Marcelo Caggiani Luizelli

(Unipampa)

Assinado eletronicamente por **Fábio Diniz Rossi**, **Usuário Externo**, em 15/12/2022, às 13:50, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **Gustavo Freitas Sanchez**, **Usuário Externo**, em 15/12/2022, às 14:31, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **Daniel Munari Vilchez Palomino**, **Usuário Externo**, em 16/12/2022, às 10:59, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **MARCELO CAGGIANI LUIZELLI**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 09/01/2023, às 09:22, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1011606** e o código CRC **841802A5**.

RESUMO

No atual estado-da-arte da codificação de vídeos está o *Versatile Video Coding* (VVC), desenvolvido com o objetivo de atender a necessidade cada vez maior por compressão de dados de vídeo decorrente do aumento do consumo desses conteúdos, além disso, traz no seu nome a versatilidade, com a proposta de suportar diversos conteúdos de vídeo incluindo as novidades que estão surgindo no mercado, como realidade virtual, realidade aumentada, vídeos em 360º entre outros, possibilita ainda suporte para altas resoluções além da 4k UHD. Em 2015 um grupo de *experts* em codificação de vídeo chamado *Joint Video Exploration Team* (JVET) se uniram para criação de um novo padrão de codificador de vídeo. Então em julho de 2020 o novo padrão de codificação VVC foi finalizado buscando alcançar uma taxa de redução de bits de 50% sobre seu antecessor *High Efficiency Video Coding* (HEVC). VVC foi implementado baseado no seu antecessor HEVC, do qual herdou a maioria das ferramentas, das quais muitas foram aprimoradas e outras novas foram adicionadas, isso possibilitou uma redução na taxa de bits média em relação ao seu antecessor de 44%. Porém esses aprimoramentos vieram com uma consequência, o aumento da complexidade computacional o que dificulta que esse novo padrão possa ser utilizado em determinadas situações como transmissão de vídeo em tempo real ou utilizado em dispositivos portáteis que possuam capacidade de energia limitada. Umas das novidades, que possui um alto custo computacional, implementadas no VVC é a *Affine*, seu propósito é realizar a predição mais eficiente de movimentos complexos como zoom, rotação e cisalhamento, que por utilizar técnicas mais sofisticadas acaba aumentando o tempo para codificar esses movimentos, podendo chegar a representar 54,75% do tempo total da *Motion Estimation* (ME). Pensando na redução do tempo de codificação, são apresentadas duas propostas para reduzir o esforço computacional do VVC, uma proposta baseada em hardware e outra em otimização de software. A proposta baseada em hardware apresenta uma heurística configurável para a etapas de Estimação de Movimento Unidirecional, Bidirecional e *Affine* do VVC. A heurística é de fácil implementação em hardware e se baseia na avaliação adaptativa de tamanhos de unidade de codificação (CU) para reduzir o consumo de energia. Três pontos de operação são suportados, alcançando um redução entre 7,7% e 30% do consumo de energia, com uma perda da qualidade de imagem entre 0,04% e 1%. A proposta baseada em software apresenta uma Estimação de Movimento *Affine* (AME) rápida para o VVC. Essa abordagem utiliza Aprendizado de Máquina com o método classificador Florestas Aleatórias (RF). O trabalho desenvolve um modelo de RF para cada tamanho de CU. Os modelos são treinados utilizando informações extraídas das CU atual, pai e vizinhas. Casa modelo é utilizado para definir se determinado tamanho de CU deverá ser ignorada pela AME ou não. A solução proposta alcança uma redução média tempo de 20% na AME e 3% no tempo total de codificação, com uma perda de qualidade de imagem de 0,07%.

Palavras-chave: VVC, Inter-quadros, Affine, Machine Learning, Random Forest

ABSTRACT

The current state-of-the-art of video coding is Versatile Video Coding (VVC), which developed the purpose of the growing need for video data compression resulting from the increased consumption of these contents. In addition, it brings versatility, with the proposal to support various video contents, including the news that is emerging in the market, such as virtual reality, augmented reality, and 360º videos, among others, also enabling support for high resolutions beyond 4k UHD. In 2015 a group of video encoding experts called the Joint Video Exploration Team (JVET) came together to create a new video encoder standard. So in July 2020, the new VVC encoding standard was finalized, aiming to achieve a 50% bitrate reduction over its predecessor High-Efficiency Video Coding (HEVC). VVC was implemented based on its predecessor HEVC, from which it inherited most of the tools, some were improved, and others added, enabling a reduction in the average bitrate compared to its predecessor of 44%. However, these improvements came with a consequence, the increase in computational complexity, which makes it impossible for this new standard to be used in certain situations, such as real-time video transmission or used in portable devices that have limited energy capacity. Affine is one of the new tools implemented in VVC to make a more efficient prediction of complex movements such as zooming, rotation, and shearing. However, the tool uses more sophisticated techniques, increasing the time to encode these movements, which may represent 54.75% of the total Motion Estimation (ME) time. Two proposals are presented, aiming at reducing the computational effort of the VVC, one hardware-friendly and the other in software optimization. The hardware-friendly proposal presents a configurable heuristic for the Unidirectional, Bidirectional, and Affine Motion Estimation steps of the VVC. The hardware-friendly heuristic is based on the adaptive evaluation of CU sizes to reduce power consumption. Three operation points are supported, achieving a reduction of between 7,7% and 30% in energy consumption, with a loss of image quality between 0,04% and 1%. The software-based proposal presents a fast Affine Motion Estimation (AME) for the VVC. This approach uses Machine Learning with the Random Forests (RF) classifier method, developing an RF model for each CU size. Information extracted from the current, parent, and neighbors CUs is used to train the models. Each model is used to define whether a given CU size should be ignored by AME or not. The proposed solution achieves an average time reduction of 20% in the AME and 3% in the total encoding time, with an image quality loss of 0,07%.

Keywords: VVC, Inter-frames, Affine, Machine Learning, Random Forest

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ALF	Adaptive Loop Filter
AMC	Affine Motion Compensation
AME	Affine Motion Estimation
AMM	Affine Motion Model
AMVP	Advanced Motion Vector Prediction
AMVR	Adaptive Motion Vector Resolution
AVC	Advanced Video Coding
BD	Bjøntegaard-Delta
BT	Binary Tree
BTH	Binary Tree Horizontal
BTV	Binary Tree Vertical
BTT	Binary-Ternary Tree
CABAC	Context-Adaptive Binary Arithmetic Coding
CPMV	Control Point Motion Vector
CTC	Common Test Conditions
CTU	Coding Tree Unit
CU	Coding Unit
CP	Control Point
CPMV	Control Pointer Motion Vector
DBF	Deblocking Filter
DT	Decision Tree
FIFO	Fisrt in First Out
FHD	Full High Definition
FPS	Frames por segundo
HEVC	High Efficiency Video Coding

HD	High Definition
HMVP	History-based Motion Vector Prediction
ICIP	IEEE International Conference in Image Processing
JEM	Joint Video Exploration Test Model
JVET	Joint Video Experts Team
JVET	Joint Video Exploration Team
LEAP	Learning-based Affine Prediction
LFNST	Low-Frequency Non-Separable Transform
LGBM	Light Gradient Boosting Machine
MC	Motion Compensation
MCP	Motion Compensation Prediction
ME	Motion Estimation
ML	Machine Learning
MTS	Multiple Transform Selection
MV	Motion Vector
MVD	Motion Vector Difference
MPEG	Moving Picture Experts Group
MSE	Mean Square Error
MTT	Multi-Type Tree
PMVP	Pairwise average Motion Vector Prediction
PSNR	Peak-To-Signal Noise Ratio
PU	Prediction Unit
QT	Quaternary Tree
QTMT	Quadtree with nested multi-type tree
RF	Random Forest
RD	Rate-Distortion

SAO	Sample Adaptive Offset
SD	Standard Definition
SDR	Standard Dynamic Range
TT	Ternary Tree
TTH	Ternary Tree Horizontal
TTV	Ternary Tree Vertical
TU	Transform Unit
UHD	Ultra-High Definition
VCEG	Video Coding Experts Group
VTM	VVC Test Model
VVC	Versatile Video Coding

SUMÁRIO

1 INTRODUÇÃO

O processamento de imagens digitais teve suas primeiras aplicações em 1920, sendo utilizadas na indústria jornalística, o que permitiu que uma imagem atravessasse o Oceano Atlântico em menos de 3 horas. A partir desse momento, juntamente com o surgimento de novas ferramentas e tecnologias para processamento de imagens, foi possível manipular uma quantidade cada vez maior de imagens com mais rapidez e eficiência (GONZALEZ; WOODS, 2009).

Um vídeo digital é formado pela exibição de uma sequência de imagens em um intervalo de tempo regular. A frequência de exibição das imagens é o que permite a sensação de movimento dos objetos em um vídeo, para que a troca de imagens não seja percebida pelo usuário, a frequência mínima de exibição deverá ser de 24 imagens por segundo, também conhecido como 24 *frames* por segundo (FPS). Como os vídeos necessitam de uma grande quantidade de imagens, a cada segundo são necessárias no mínimo 24 imagens, consequentemente produzem um volume muito elevado de dados. Por exemplo, um vídeo colorido de 150 minutos, com 30 *frames* por segundo e com resolução *Standard Definition* (SD de 720×480 *pixels*), geraria um volume de dados de 280GB, que por sua vez, devem ser armazenados ou transmitidos. Para que isso seja viável, os dados necessários para representar o vídeo devem ser reduzidos.

Com o objetivo de reduzir esse volume de dados, entre os anos de 1970 e 1980 surgiram os primeiros conceitos de padrões de codificação de vídeo. Os padrões de codificação possuem a capacidade de comprimir os dados gerados pelos vídeos (codificar) e posteriormente restaurar esses dados para que o vídeo possa ser exibido (decodificar), podendo ser chamados também de CODEC de vídeo (CO - *COding* e DEC - *DECoding*) (ZHANG; MAO, 2019).

Ao longo do tempo, com o surgimento de novos dispositivos capazes de manipular diferentes conteúdos de vídeos digitais e novas tecnologias com maior capacidade de processamento de dados, foram emergindo também novos padrões de codificação de vídeo, atendendo a demanda pela compressão dos dados e mantendo a melhor qualidade de imagem possível.

No cenário atual os vídeos fazem parte do nosso dia-a-dia, eles estão presente no lazer, trabalho, estudo, segurança, entre outros, isso ocasionou um crescente aumento no volume de dados gerados. Alguns pontos que contribuíram para esse cenário podem ser destacados: a maior facilidade de acesso de usuários à dispositivos com capacidade de capturar, armazenar e transmitir vídeos digitais em altas resoluções, como *smartphones*, câmeras digitais, óculos de realidade virtual, aviação não tripulada (drones); o surgimentos de novos conteúdos de vídeos como realidade aumentada, vídeos imersivos, compartilhamento de telas, vídeos em 360º, exigindo cada vez mais qualidade e maiores resoluções de imagem; o aumento da capacidade de transmissão e velocidade da internet, a facilidade de acesso a redes de banda larga, permitiram a utilização cada vez maior de ferramentas que

possibilitam o acesso e compartilhamento de vídeos, tais como Facebook, Netflix, Youtube, jogos online, vídeo conferência, entre outros.

Outro evento inesperado que também contribuiu para o aumento do consumo de vídeo foi a pandemia de COVID-19, onde as pessoas precisaram ficar em casa e acabaram utilizando as diversas ferramentas disponíveis para realizar suas atividades diárias de trabalho, lazer e estudos, como reuniões de trabalho, aulas online, jogos online, *streaming* de vídeo, sendo essa última a que mais contribuiu com esse aumento, representando 57,64% de todo tráfego em 2020, tendo o Youtube a maior representatividade nesse percentual com 15,94% (SANDVINE, 2020). Segundo relatório da (SANDVINE, 2021), aponta que serviços de *streaming* de vídeo também representa o maior percentual do tráfego de dados nas redes móveis, representando 48,9% do *downstream* e 19,4% do *upstream*, onde o Youtube também se mantém como a aplicação de *streaming* de vídeo que mais gera tráfego, representando 20% do *downstream* e 4,2% *upstream*, seguido pelo Facebook Vídeo com 11,3% do *downstream* e 4,2% de *upstream*.

Todos os pontos descritos anteriormente também tem contribuído significativamente para o aumento do tráfego de dados de vídeo na rede (BROSS et al., 2021b). Estima-se que 80% do tráfego global de dados da internet seja referente a transmissão de vídeos digitais (CISCO, 2019), e isso tende a aumentar, pois segundo (CISCO, 2020), há uma estimativa de aumento de 33% em 2018 para 66% em 2023 do número de TVs residenciais com suporte a resolução *Ultra-High Definition* (UHD) 4k (3840×2160 pixels), isso geraria um aumento significativo de dados na rede, uma vez que o *bitrate* da resolução 4k é entre 15 e 18kbps, sendo o dobro da *High Definition* (HD 1280×720 pixel) e nove vezes maior que a *Standard Definition* (SD).

A Figura 1, apresenta uma comparação entre as resoluções Full HD, 4k, 5k, 8k e 10k, onde é possível observar as proporções entre as resoluções e também a quantidade de *pixels* empregada em cada uma delas.

Diante desse cenário, a indústria e a academia têm investido fortemente em pesquisas para o desenvolvimento de ferramentas cada vez mais eficientes para realizar a compressão de vídeos e reduzir a quantidade de dados gerada. Foi então proposta a nova geração do padrão de codificadores de vídeo o *Versatile Video Coding* (VVC), conhecido também como padrão H.266.

Em 2015 um grupo de experts em codificação de vídeo chamado *Joint Video Exploration Team* (JVET), composto por ITU-T *Video Coding Experts Group* (VCEG) e a ISO/IEC *Moving Picture Experts Group* (MPEG) se juntaram para explorar novas tecnologias que melhorassem a eficiência da codificação. Dois anos mais tarde em 2017 as atividades resultaram no *Joint Video Exploration Test Model* (JEM) que já apresentava um redução de mais de 30% na taxa de bits comparado com o padrão antecessor o *High Efficiency Video Coding* (HEVC) (SULLIVAN et al., 2012), esses resultados mostraram que seria viável a continuidade do novo padrão. O nome do grupo de *experts* mudou para

Figura 1 – Comparação entre as resoluções Full HD, 4k, 5k, 8k e 10k.

Fonte: (SIGNATURE, 2022)

Joint Video Experts Team (JVET), e em abril de 2018 foi iniciado o projeto oficial do VVC (BROSS et al., 2021b). Em julho de 2020 o novo padrão de codificação VVC foi finalizado buscando alcançar uma taxa de redução de *bits* de 50% sobre seu antecessor HEVC (BROSS et al., 2021a).

Para o VVC alcançar a eficiência de codificação esperada, muitas ferramentas foram herdadas do seu antecessor HEVC, e outras foram adicionadas ou aprimoradas, contudo esse conjunto de modificações que permitiu a redução da taxa de bits em aproximadamente 44,4% quando comparado com seu antecessor, teve como consequência o aumento no custo computacional, ou seja, utilizar mais tempo de processamento para codificar um vídeo. O aumento desse custo computacional tornou o tempo de codificação do VVC 10 vezes mais lento que o HEVC, o que atualmente torna inviável seu uso em alguns cenários como aplicações de transmissão de vídeos de altas resoluções em tempo real (SIQUEIRA; CORREA; GRELLERT, 2020) (BRANDENBURG et al., 2020). Outro ponto observado é que o volume de dados de vídeo também dominam as redes móveis, o que aponta para a grande utilização de dispositivos portáteis para manipulação de vídeos. Sabendo que esses dispositivos possuem capacidade limitada de energia, a redução da complexidade computacional do codificador se torna primordial para sua utilização em larga escala em dispositivos portáteis. Em toda essa dissertação, os termos complexidade, esforço e custo computacional são definidos como o tempo gasto na codificação.

Levando em conta todas as colocações feitas anteriormente, pesquisas que busquem soluções para a redução da complexidade do codificador sem diminuir significativamente a qualidade do vídeo são importantes.

1.1 Motivação

Quando o padrão de codificação de vídeo VVC foi criado, a prioridade inicial foi com a redução da taxa de *bits* mantendo a qualidade do vídeo sem muita preocupação com o custo computacional, pois como mencionado anteriormente o VVC chega a ser até 10 vezes mais lento que o HEVC (SIQUEIRA; CORREA; GRELLERT, 2020). Neste sentido, reduzir o custo computacional é um tópico importante na área de codificação de vídeo, uma vez que o codificador deve se adaptar a vários cenários, como codificações que priorizem mais a qualidade sem muita preocupação com o custo computacional (e.g. aplicações de segurança), até codificações onde o custo computacional da codificação é importante (e.g. em sistemas embarcados), permitindo perdas aceitáveis na qualidade, como transmissões em tempo real e implementação em dispositivos portáteis.

Uma das etapas mais lentas da codificação de vídeo é a etapa de predição inter-quadros, onde são aplicadas as ferramentas de Estimação de Movimento (em inglês *Motion Estimation* - ME) e a etapa de Compensação de Movimento (em inglês *Motion Compensation* - MC) responsáveis por explorar a redundância temporal entre os *frames* de um vídeo. Embora seja uma das etapas mais custosas da codificação de vídeo, no VVC ainda houve um aumento de sua complexidade, ao ser introduzida uma nova ferramenta, a Estimação de Movimento *Affine* (em inglês *Affine Motion Estimation* - AME). A AME é responsável por realizar a estimação de movimentos não translacionais como zoom, rotação e cisalhamento, auxiliando o padrão a ampliar seus ganhos em compressão. Entretanto, essa nova ferramenta apresenta um grande custo computacional, podendo representar em média 54,75% do tempo total da ME (PARK; KANG, 2019).

Nesse sentido, o trabalho propõe duas soluções para reduzir a complexidade do VVC, uma das propostas é baseada em hardware e a outra baseada em otimização de software. A solução baseada em hardware propõe uma heurística configurável, focando nas etapas da ME Unidirecional, Bidirecional e *Affine*, que permite selecionar diferentes níveis de economia de energia de acordo com o ponto de controle e a resolução do vídeo. A solução baseada em software aplicará técnicas de Aprendizado de Máquina (em inglês *Machine Learning* - ML) com o método Classificador Florestas Aleatórias (em inglês *Random Forest Classifier* - RFC) focando na etapa ME *Affine*. Nessa proposta foi criado um modelo RFC para cada tamanho de Unidade de Codificação (em inglês *Coding Unit* - CU), evitando que determinados tamanhos de CU não sejam processados pela AME, quando provavelmente a AME não seria selecionada como o melhor modo de predição para esse tamanho de CU.

1.2 Objetivo

Como mencionado anteriormente um dos impedimentos para que o VVC possa ser utilizado em larga escala é o elevado custo computacional que ele apresenta. Então, esse trabalho tem como objetivo principal reduzir a complexidade computacional do VVC

focando nas etapa da ME *Affine*. Para alcançar esse objetivo duas soluções são propostas, uma baseada em hardware propondo uma heurística configurável para selecionar diferentes níveis de economia de energia e a outra baseada em otimização de software utilizando técnicas de Aprendizado de Máquina com o método Classificador Florestas Aleatórias.

Os objetivos específicos são listados abaixo:

- Apresentar as etapas de ME Unidirecional, Bidirecional e *Affine* do padrão VVC;
- Analisar do impacto da complexidade da AME na predição inter-quadros para cada tamanho de CU;
- Desenvolver soluções que diminuam a complexidade computacional do VVC;

Os dois trabalhos desenvolvidos alcançaram o objetivo de redução do esforço computacional do VVC com o foco na ME *Affine*. O trabalho propondo a redução do consumo de energia de maneira configurável, alcançou uma economia variando entre 7,7% e 30% dependendo do ponto de operação e a resolução escolhidas, com uma perda de eficiência de codificação variando entre 0,04% e 1%. Esse trabalho foi submetido e aceito no 29º IEEE *International Conference on Electronics Circuits and Systems* (ICECS), e também recebeu o prêmio de melhor pôster apresentado no 6º IEEE *Seasonal School on Digital Processing of Visual Signals and Applications* (DPVSA). O segundo trabalho, propondo a otimização da ferramenta de ME *Affine* utilizando Aprendizado de Máquina e utilizando o método classificador de Florestas Aleatórias, obteve uma redução média de tempo de 20% na AME e 3% no tempo total de codificação, com uma perda de eficiência de codificação de 0,07%. Baseado nesse trabalho um artigo foi escrito e submetido ao evento IEEE *International Symposium on Circuits and Systems* 2023 (ISCAS).

1.3 Organização do Trabalho

O trabalho está organizado em sete capítulos. No Capítulo 2 possui o referencial teórico, onde são abordados os seguintes assuntos: conceitos do codificador de vídeo genérico com foco maior no processo de predição inter-quadro; padrão de codificação VVC detalhando as etapas de particionamento de blocos, predição inter-quadros no VVC e Estimação de Movimento *Affine*. O capítulo 3 apresenta os conceitos de Aprendizado de Máquina incluindo Árvores de Decisão e Florestas Aleatórias. No capítulo 4 são apresentados os trabalhos relacionados e uma análise entre eles. O Capítulo 5 apresenta a metodologia utilizada no trabalho, já o Capítulo 6 apresentada a proposta baseada em hardware e no Capítulo 7 é apresentada a proposta baseada otimização de software. Para finalizar, no Capítulo 8 são apresentadas as conclusões e trabalhos futuros.

2 REFERENCIAL TEÓRICO

Neste capítulo é apresentado os conceitos de um codificador de vídeo, necessários para o desenvolvimento do trabalho. Inicialmente é abordado o conceito de um codificador de vídeo genérico com foco na predição inter-quadros, na sequência o padrão de codificação VVC é detalhando, descrevendo suas principais melhorias com foco maior nas etapas de particionamento de bloco e predição inter-quadros.

2.1 Codificador de Vídeo

O principal objetivo de um codificador de vídeo é reduzir ou comprimir o número de *bits* necessários para representar um vídeo. Um dispositivo ou programa que realiza a compressão de um sinal de vídeo é chamado de *encoder* e um dispositivo ou programa que descompacta esse sinal de vídeo é chamado de *decoder*, um par de *enCOder/DECoder* é chamado de CODEC (RICHARDSON, 2002). Quando ocorre o processo de decodificação do sinal de vídeo, o resultado pode ser idêntico ao original (sem perdas) ou pode sofrer alguma degradação (com perdas).

Um sinal de vídeo pode possuir até três tipos de redundância: estatística, psico-visual e de codificação. A redundância estatística ocorre devido alguns padrões de dados serem mais prováveis de ocorrerem que outros, essa redundância está ligada a alta correlação entre *pixels* vizinhos, podendo ser uma correlação espacial (intra-quadro) dentro da própria imagem ou temporal (inter-quadro) onde ocorre uma correlação entre o quadro atual e quadros vizinhos. Psico-visual explora o comportamento do sistema visual humano (HVS - em inglês *Human Visual System*), pois ele é mais sensível a informações visuais com baixas frequências do que a informações com altas frequências, podendo assim, ser eliminadas algumas informações menos sensíveis ao HVS. A redundância de codificação utiliza símbolos para representar as informações, onde símbolos mais curtos são utilizados para representar informações que ocorrem com maior frequência. Um codificador de vídeo explora essas redundâncias para reduzir o volume de dados (CHAKRABARTI; BATTA; K., 2015).

Algumas dessas redundâncias podem ser percebidas na Figura 2, onde várias regiões da mesma imagem possuem enormes similaridades, como as rodovias, áreas verdes e o corpo do tucano. Essa similaridade pode ser explorada pelo codificador, armazenando um único bloco e fazendo referência aos demais que possuem maior similaridade, podendo então eliminar os blocos redundantes. Essa redundância é chamada de redundância espacial ou intra-quadro.

Outra redundância que é explorada pelo codificador é a que ocorre entre diferentes quadros. Como um vídeo necessita que uma grande quantidade de quadros estáticos sejam exibidos em um determinado intervalo de tempo, proporcionando a sensação de movimento, esses quadros possuem uma similaridade muito grande entre eles, pois são capturados em um intervalo de tempo muito curto. Um exemplo pode ser visto na Figura 2(a) e 2(b), onde

dois quadros gerados de uma mesma cena possuem muitos blocos similares, com apenas algumas variações na posição dos veículos e movimento do tucano, o restante da imagem permanece inalterada em relação ao outro quadro. Então os blocos com informações redundantes podem ser descartados através de uma codificação diferencial entre os quadros, onde o codificador vai referenciar no quadro atual, apenas os blocos que sofreram alguma mudança, indicando no quadro de referência onde se encontra a melhor similaridade com o bloco atual. Essa redundância é chamada de redundância temporal ou inter-quadro.

Figura 2 – Redundância espacial e temporal em quadros de um vídeo.

Fonte: Elaborado pelo autor.

Para analisar se o resultado da codificação é satisfatório, a qualidade de um vídeo pode ser mensurada de duas maneiras: qualidade subjetiva e qualidade objetiva. Mensurar a qualidade subjetiva de um vídeo é bastante difícil e imprecisa uma vez que a qualidade visual humana pode ser influenciada por diversos fatores, como maior sensibilidade a detalhes de luminância do que de detalhes de cor, maior sensibilidade a altos contraste do que a baixos, entre outros, que podem levar a alterações no resultado da avaliação. Para esse tipo de avaliação é recomendado que seja feita por um avaliador com experiência nessa área (RICHARDSON, 2002). Entretanto, essa avaliação é considerada demorada e com um custo elevado devido a necessidade de um avaliador humano.

Para contornar esse problema foi desenvolvido um sistema para mensurar a qualidade visual de um vídeo de maneira objetiva, ele não substitui a avaliação subjetiva, porém é mais fácil de ser aplicado e é utilizado largamente no desenvolvimento de codificadores e para propósitos de comparação de qualidade de um vídeo durante a compressão e descompressão (RICHARDSON, 2002).

Para se obter o resultado de uma avaliação objetiva, a métrica mais utilizada é o *peak-to-signal noise ratio* (PSNR), que é medida em escala logarítmica e expressa em decibéis (dB) (RICHARDSON, 2002). Além disso, essa métrica é baseada no erro quadrático médio (em inglês *Mean Square Error - MSE*) entre o quadro original e o quadro codificado. As equações 2.1 e 2.2 mostram as definições do MSE e do PSNR, respectivamente, onde R é uma matriz que representa as amostras do quadro reconstruído,

O representa as amostras do quadro original, e h e w diz respeito a altura e largura do quadro, respectivamente. Na equação 2.2, MAX representa o maior valor de uma amostra.

$$MSE(x,y) = \sum_{i=0}^{h-1} \sum_{j=0}^{w-1} (R_{i,j} - O_{i,j})^2 \tag{2.1}$$

$$PSNR_{db} = 10log_{10} \frac{MAX^2}{MSE} \tag{2.2}$$

O processo de codificação de um quadro é mostrado pela Figura 3, que apresenta o esquema de um codificador de vídeo genérico. Como entradas para o codificador temos os quadro atual e os quadros de referência. O quadro atual é o quadro codificado no momento, os quadros de referência são quadros já previamente codificados e reconstruídos para serem utilizados no processo de predição do quadro atual e posteriores.

A primeira etapa pela qual o quadro atual irá passar é seu particionamento, onde ele é dividido em blocos (SZE; BUDAGAVI; SULLIVAN, 2014), e então prosseguirá para a etapa de predição, podendo ser a predição intra-quadros, que explora a redundância espacial dentro do próprio quadro como mencionado anteriormente, ou a predição inter-quadros que explora a redundância temporal que ocorre entre quadros diferentes, ou seja, entre o quadro atual e um quadro de referência/reconstruído. Ainda na predição inter-quadros temos as etapas de Estimação de Movimento e Compensação de Movimento, responsáveis por identificar o movimento entre os quadros. Finalizado o processo de predição intra ou inter-quadro, o quadro gerado por esse processo é subtraído do quadro atual resultando um quadro residual, contendo apenas informações referente às diferenças entre o quadro atual e o reconstruído (RICHARDSON, 2002).

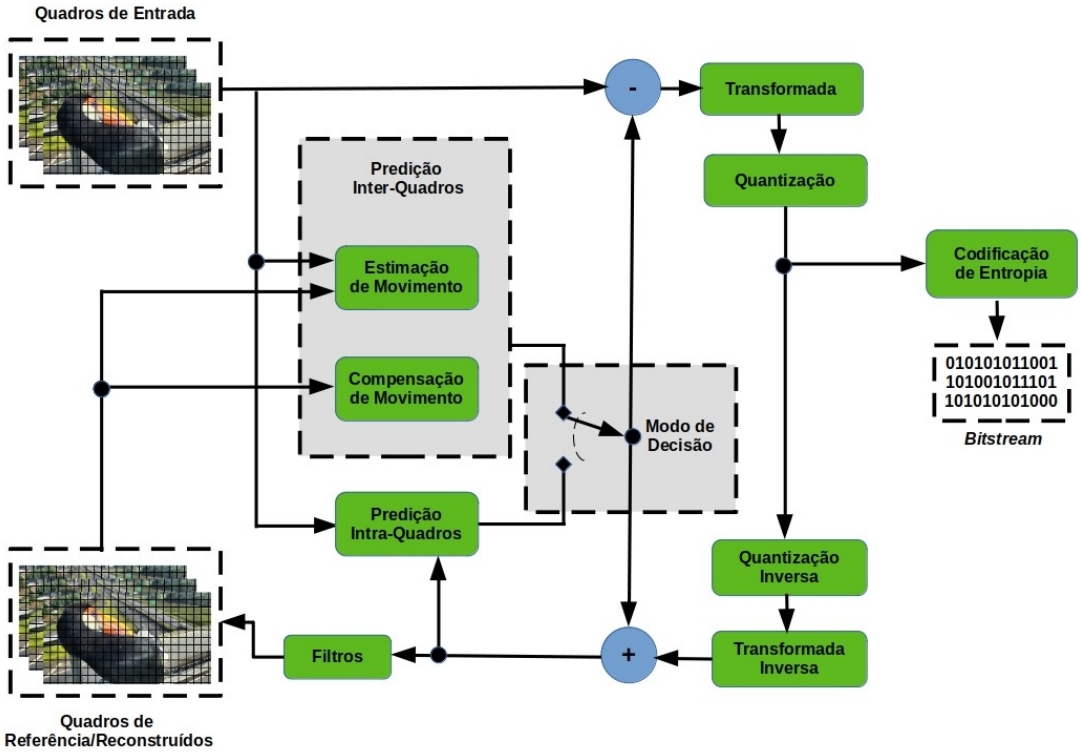
As informações residuais irão seguir para a etapa de Transformada (RICHARDSON, 2002), onde a imagem é convertida de um domínio espacial para um domínio de frequências, sendo possível identificar regiões com altas frequências, que são menos sensíveis ao HVS. Até essa etapa as informações ainda não sofreram compressão.

Na etapa de Quantização (RICHARDSON, 2002) às frequências menos sensíveis ao HVS, geradas pela etapa de Transformada irão ser cortadas/atenuadas, uma vez que essas frequências não influenciam ou influenciam muito pouco na percepção do espectador. Nessas duas etapas (transformada e quantização) são exploradas a redundância psico-visual e ocorre a primeira redução nos dados com perdas de informação.

Na etapa de Entropia (RICHARDSON, 2002) é explorada a redundância de codificação. Após a etapa de transformada e quantização, é realizada uma análise estatística sobre os valores resultantes para identificar a frequência que ocorre cada valor, os valores que apresentarem maiores ocorrências são representados por símbolos com uma quantidade menor de *bits*, reduzindo o montante de *bits* final. Nessa etapa é gerado o resultado final do codificador, ou seja, o vídeo comprimido ou também chamado de *bitstream*, podendo ser transmitido ou armazenado.

Além do processo de codificação de vídeo, o codificador também realiza o processo de decodificação através das etapas de Quantização e Transformada inversa. O quadro decodificado/reconstruído é usado como quadro de referência.

Figura 3 – Estrutura de um codificador de vídeo genérico.



Fonte: Elaborado pelo autor.

Nessa seção foi apresentado os conceitos básicos de um codificador de vídeo e todo o processo executado para realizar a compressão dos dados de um vídeo. Na próxima seção será abordada a predição inter-quadros, etapa importante para a dissertação, assim ela será melhor detalhada.

2.2 Predição Inter-Quadros

Nessa seção é apresentada a predição inter-quadros de forma genérica, na seção 2.3.2, ela é detalhada no contexto do codificador VVC.

Um vídeo é um conjunto de imagens/quadros estáticos exibidos sequencialmente em um determinado intervalo de tempo. Como mencionado anteriormente a predição inter-quadros explora a correlação entre quadros temporalmente localizados em um vídeo, ou seja, redundância temporal.

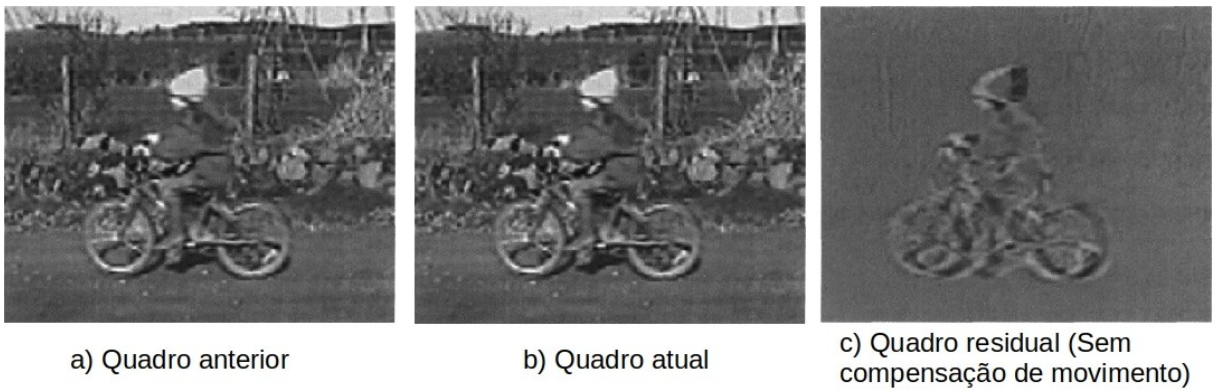
O processo de compressão é alcançado por duas funções principais: Predição e Compensação. A predição cria uma predição do corrente quadro baseado em um ou

mais quadros previamente codificados. Na compensação ocorre a subtração do quadro anterior/predito pelo quadro corrente para construir o quadro residual.

A saída do processo de compensação é um quadro residual, como mostrado na Figura 4. A etapa chave desse processo é a predição, pois quanto melhor for o processo de predição menor é a energia residual, e melhor é o resultado da compressão.

Na Figura 4(c), é mostrado o quadro residual, resultante da subtração do quadro de referência/anterior Figura 4(a), pelo quadro atual Figura 4(b). Na Figura 4(c) os tons de cinza médio representam valor residual zero, e áreas mais claras e escuras indicam um valor residual positivo ou negativo respectivamente, indicando as regiões de movimento ou diferença entre os quadros. Observa-se que quanto mais valores residuais iguais a zero (área cinza claro) como informação residual, maior é a taxa de compressão.

Figura 4 – Processo de compensação entre quadros.



Fonte: (RICHARDSON, 2002)

2.2.1 Predição de Compensação de Movimento

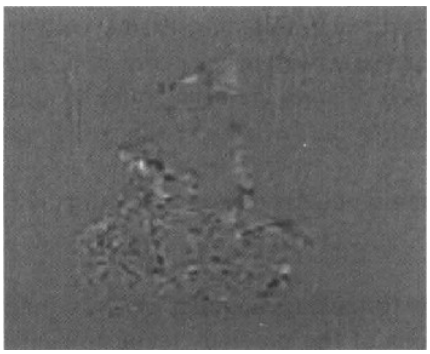
A compensação proporciona uma melhor compressão quando codifica sucessivos *frames* com muita similaridade, mas não funciona adequadamente quando ocorrem significativas mudanças entre o quadro atual e o quadro referência. Um melhor desempenho pode ser alcançado usando técnicas de estimação de movimento (em inglês *Motion Estimation* - ME) e compensação de movimento (em inglês *Motion Compensation - MC*).

Mudanças entre os quadros ocorrem principalmente pelo deslocamento de objetos dentro da cena. Para detectar esse deslocamento de objetos que ocorrem entre o quadro de referência e o quadro atual, o codificador utiliza um modelo para estimar esse movimento, onde uma região do quadro, frequentemente um bloco retangular (por exemplo 16×16 *pixels*), é comparado com os blocos de um quadro previamente codificado (quadro de referência), buscando encontrar o bloco que retorne o melhor valor residual, ou seja, o valor mais próximo a zero. Esse processo é chamado de ME.

Então o codificador utiliza as informações de movimento geradas pela ME, para reconstruir o quadro predito, essas informações de movimento normalmente são representadas pelos vetores de movimento (em inglês *Motion Vector* - MV). Então o quadro predito, que é um quadro reconstruído baseado nas informações de movimento, é subtraído do quadro atual gerando o quadro residual. Esse processo se chama MC. A união da ME com a MC é chamada de predição de compensação de movimento (em inglês *Motion Compensation Prediction* - MCP).

O quadro residual gerado através da MC contém mais resíduos zero ou próximos a zero, que o quadro gerado sem MC, um exemplo pode ser observado na Figura 5 contendo um quadro residual gerado utilizando a MC. Se comparado com o quadro residual da Figura 4(c) (sem MC), o quadro gerado pela MCP apresenta uma nítida redução das áreas cinza escuro e cinza claro que representam as diferenças entre os blocos, assim com uma maior similaridade entre os blocos, menos dados residuais são gerados e maior é a taxa compressão. Porém essa eficiência em codificação da MC vem acompanhada de um custo computacional mais elevado devido aos algoritmos de busca utilizados para encontrar os melhores blocos (RICHARDSON, 2002).

Figura 5 – Quadro residual obtido utilizando processo de compensação de movimento.



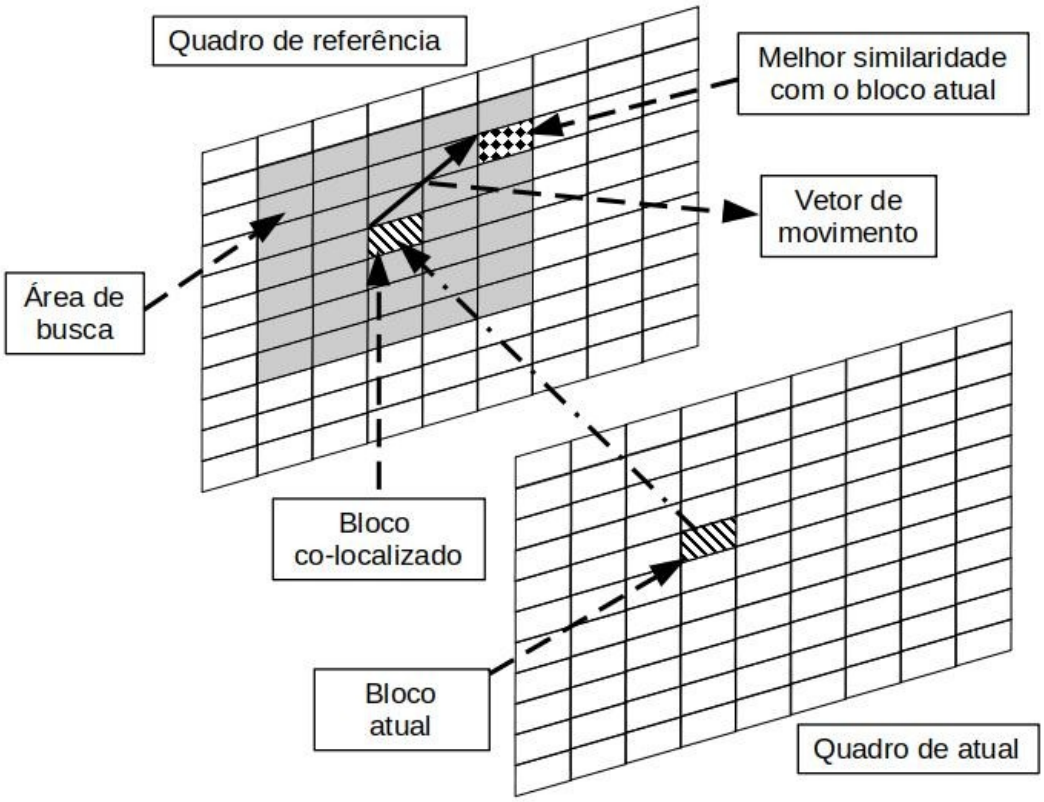
Fonte: (RICHARDSON, 2002)

A ideia inicial para encontrar o bloco mais similar no quadro de referência, seria comparar o bloco do quadro atual com todos os blocos do quadro de referência, essa comparação é chamada de *Full Search*, porém isso seria inviável devido ao alto custo computacional, pois vários cálculos devem ser realizados para encontrar o melhor resultado residual. Então para solucionar esse problema foram desenvolvidos algoritmos de busca rápida, chamados de *Fast Search*. Esses algoritmos definem uma áreas de busca próxima a localização do bloco a ser predido, áreas onde estatisticamente há maiores chances de encontrar a melhor correlação entre os blocos, reduzindo o custo computacional da busca.

A Figura 6 apresenta um esquema de como ocorre a busca inter-quadros. Nela é possível ver no quadro de referência o bloco co-localizado representado por listras, e ao seu redor a área de busca representada pela região em cinza. Nessa região o codificador

irá pesquisar pelo bloco que possuir maior similaridade com o bloco atual, ao encontrar o bloco mais similar, representado na imagem pelo bloco quadriculado, um MV é gerado, o qual descreve o movimento do bloco do quadro atual em relação ao bloco escolhido no quadro de referência. O MV é representado por duas dimensões (x,y) indicando o deslocamento horizontal e vertical do bloco, respectivamente.

Figura 6 – Esquema de busca inter-quadros.



Fonte: Elaborado pelo autor.

Nessa seção foi apresentado o modo de predição inter-quadro, mostrando como é realizado o processo que explora as redundâncias temporais entre os quadros. Como pode ser percebido no texto, essa etapa é importante para a compressão do vídeo, pois busca eliminar as similaridades entre os quadros, na contramão, apresenta um custo computacional elevado em virtude da necessidade das várias comparações entre os blocos.

2.3 Padrão de Codificação de vídeo *Versatile Video Coding* - VVC

Até o lançamento oficial do *Versatile Video Coding* (VVC) em 2020, dois padrões haviam sido desenvolvidos anteriormente pelas instituições ITU-T Video *Coding Experts Group* (VCEG) e o ISO/IEC *Moving Picture Experts Group* (MPEG), o padrão *Advanced*

Video Coding (AVC) (MARPE; WIEGAND; SULLIVAN, 2006) e o *High Efficiency Video Coding* (HEVC).

Em 2013, a primeira versão do HEVC alcançou uma redução da taxa de bits de 50% sobre seu antecessor o padrão AVC. Agora os esforços estão voltados para o padrão estado-da-arte o VVC, para alcançar essa mesma redução de 50% na taxa de *bits* em relação ao seu antecessor o HEVC (BROSS et al., 2021a). Trabalhos como de (SIQUEIRA; CORREA; GRELLERT, 2020) apontam que o padrão VVC já alcança uma taxa de redução de *bits* em torno de 44% em relação ao HEVC.

No cenário atual os vídeos fazem parte do nosso dia-a-dia, eles estão presente no lazer, trabalho, estudo, segurança, entre outros, isso ocasionou um crescente aumento no volume de dados gerados. Alguns pontos contribuíram para esse cenário como a facilidade de acesso a dispositivos de manipulação de vídeo, como por exemplo celulares, câmera fotográficas, TVs de alta resolução; a popularização de novos conteúdos de vídeo, como compartilhamento de conteúdos de tela, vídeos em 360º, realidade virtual, vídeos em imersividade; o aumento da velocidade e largura de banda da internet, que proporcionou maior facilidade para consumo e compartilhamento de vídeos. Estima-se que 80% dos dados que trafegam pela internet sejam referente a vídeos, cenário agravado ainda mais durante a pandemia de COVID-19.

Devido aos fatores listados acima, surge a necessidade de taxas de compressão maiores que as suportadas pelo padrão atual o HEVC. Assim foi proposta a nova geração do padrão de codificadores de vídeo o *Versatile Video Coding* - VVC, conhecido também como H.266. Em 2015 um grupo de *experts* em codificação de vídeo chamado *Joint Video Exploration Team* (JVET), composto por ITU-T *Video Coding Experts Group* (VCEG) e a ISO/IEC *Moving Picture Experts Group* (MPEG) se juntaram para explorar novas tecnologias que melhorassem a eficiência da codificação.

Dois anos mais tarde, em 2017, as atividades resultaram no *Joint Video Exploration Test Model* (JEM) que já demonstrava mais de 30% na redução da taxa de *bits* comparado com o padrão antecessor o *High Efficiency Video Coding* (HEVC) (SZE; BUDAGAVI; SULLIVAN, 2014), esses resultados mostraram que seria viável a continuidade do novo padrão. O nome do grupo de *experts* mudou para *Joint Video Experts Team* (JVET), e em abril de 2018 foi iniciado o projeto oficial do VVC (BROSS et al., 2021b). Em julho de 2020 o novo padrão de codificação VVC foi finalizado (BROSS et al., 2021a).

A JVET disponibiliza também um software de referência do padrão VVC, que traz todas as ferramentas do codificador implementadas, permitindo que alterações e testes possam ser realizados. O software é chamado de Modelo de Teste VVC (em inglês *VVC Test Model* - VTM) (VTM, 2022).

Para alcançar a eficiência desejada o VVC foi desenvolvido seguindo o mesmo padrão de codificação híbrido utilizado pelos padrões anteriores e herdado do HEVC, de onde também herdou várias ferramentas que foram melhoradas, além disso novas

ferramentas também foram implementadas. Abaixo são apresentadas algumas dessas ferramentas e melhorias em cada uma das etapas (PAKDAMAN et al., 2020).

Etapa de Particionamento de Blocos - VVC estende o conceito de Unidade de Árvore de Codificação (em inglês *Coding Tree Unit* - CTU) do HEVC. Uma das alterações na etapa de particionamento ocorreu na maneira de realizar o particionamento dos blocos. VVC utiliza um esquema chamado Árvore Quaternária Multi-tipo (em inglês - *Quadtree with nested Multi-type Tree* - QTMT), permitindo CTU com tamanho máximo de 128×128 *pixels*, diferente do HEVC que utiliza o esquema de Árvore Quaternária (em inglês *Quaternary Tree* - QT), permitindo CTUs com tamanho máximo de 64×64 *pixels*.

Com o QTMT as CTUs podem ser divididas em blocos menores chamados de Unidades de Codificação (inglês *Coding Unit* - CU) das seguintes formas: quadrática e retangular. Na forma quadrática o bloco é dividido com dimensões de largura e altura iguais. Na foma retangular, o bloco pode ser dividido de maneira binária ou ternária, ambos podendo ser divisíveis na horizontal ou na vertical.

A QTMT também unifica os conceitos de Unidade de Codificação (CU), Unidades de Predição (em inglês *Prediction Unit* - PU) e Unidade de Transformação (em inglês *Transform Unit* - TU), todos em único conceito de CU (*Coding Unit*).

Etapa de Predição Intra - HEVC utiliza 35 modos de direção intra, VVC permite agora 67 modos, consistindo em 65 modos direcionais, um DC e um planar. O modo direcional intra é aplicado em CUs com tamanho máximo de 64×64 *pixels*. Para modos angulares, intra suporta direções de 45º a -135º. Diferente do HEVC, blocos não quadrados podem ser acessados pela predição intra no VVC, pois um esquema adaptativo é usado em modo de ângulos amplos.

Para reduzir ainda mais a redundância de componentes cruzados, VVC aplica um esquema de predição linear que permite que amostras de crominância possam ser preditas baseadas em amostras de luminância reconstruídas. Outro melhoramento é que o VVC estende amostras de referência para habilitar o uso de várias linhas de referência, melhorando a qualidade da predição.

Etapa de Predição Inter - As informações de movimento no VVC podem ser sinalizadas explicitamente através dos parâmetros de movimento, através do modo *merge skip* ou através do modo *merge*, esse último deriva parâmetros de movimento de CUs candidatas vizinhas. Os tipos de candidatos utilizados pelo modo *merge* são representados por MVs de CUs vizinhas localizadas espacialmente, temporalmente e também por vetores de movimento zero, da mesma maneira que o HEVC, porém o VVC estende esse modo, adicionando mais dois tipos de candidato, o MV médio em pares e o MV baseado em histórico de uma tabela FIFO (*Fisrt in First Out*).

VVC também insere uma nova ferramenta chamada Estimação de Movimento *Affine* (em inglês *Affine Motion Estimation* - AME) que busca melhorar o processo de predição para movimentos complexos como zoom, rotação e cisalhamento. Implementa

também os modos *merge* e Predição de Vetor de Movimento Avançado (em inglês *Advanced Motion Vector Prediction* - AMVP) para o modo *affine*, melhorando a predição e reduzindo os dados do conjunto de informações de movimento. Enquanto o HEVC usa uma precisão constante de 1/4 de amostra para sinalizar o MVD (em inglês *Motion Vector Difference*), VVC utiliza AMVR (em inglês *Adaptive Motion Vector Resolution*) que permita uma precisão entre 1/4 até 1/16 de amostra de luminância dependendo do modo escolhido.

Transformada - VVC usa blocos com tamanho máximo de 64×64 *pixels* de luminância e 32×32 *pixels* de crominância para se adequar a amostras de alta resolução. Para explorar ainda mais a redundância espacial, uma transformada secundária é introduzida para usar transformadas não separáveis em blocos 4×4 e 8×8 *pixels*.

Codificação de entropia - VVC utiliza uma versão otimizada do *Context-Adaptive Binary Arithmetic Coding* (CABAC). Desta forma, a seleção de modelos de probabilidade para elementos de sintaxe depende do valor e do número de elementos diferentes de zero em uma vizinhança local.

Filtros In-Loop - VVC utiliza três filtros In-Loop, o *Deblocking Filter* (DBF) e o *Sample Adaptive Offset* (SAO) que são muito similar ao do HEVC, já o *Adaptive Loop Filter* (ALF) é usado somente no VVC. ALF utiliza um esquema de classificação de blocos para escolher entre 25 diferentes filtros, baseados na direção e no nível dos gradientes locais.

Nessa seção foram apresentadas algumas das melhorias implementadas no VVC, nas próximas seções será discutido dois assuntos relevantes para o trabalho que são a estrutura de particionamento de blocos e predição inter-quadro implementada no VVC.

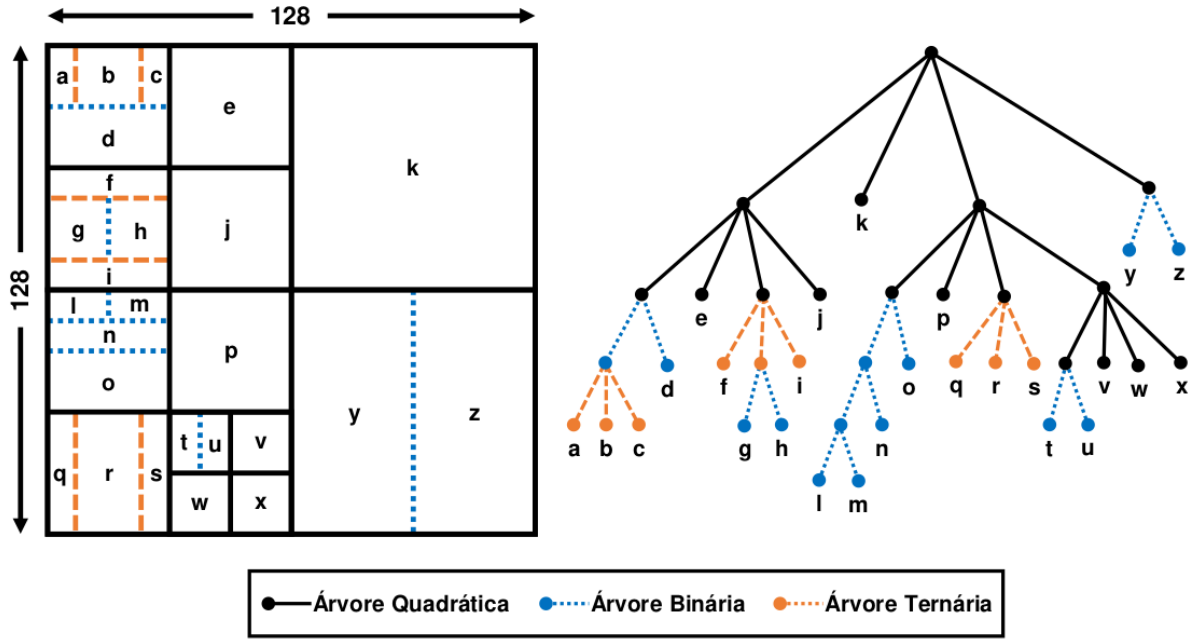
2.3.1 Particionamento de Blocos

O processo de codificação não é realizado sobre o quadro inteiro como um bloco único, o quadro é dividido em partes menores e então essas partes são processadas pelo codificador. Tanto no HEVC como no VVC o quadro é inicialmente particionado em blocos de tamanhos iguais chamados de *Coding Tree Unit* (CTU), o tamanho das CTUs é definido no início da codificação e é mantido até o final. No HEVC o tamanho máximo suportado pelas CTUs é 64×64 *pixels*, já no VVC o tamanho máximo passa a ser 128×128 *pixels*. CTUs maiores apresentam melhor eficiência de codificação quando aplicadas em quadros com altas resoluções com UHD ou superiores (HUANG et al., 2021).

HEVC utiliza uma estrutura de particionamento chamada *Quaternary Tree* (QT) que é aplicada recursivamente em cada CTU, resultando em uma ou várias CUs, todas com formato quadrado. VVC implementa a divisão de árvore binária (em inglês *Binary Tree* - BT) e divisão de árvore ternária (em inglês *Ternary Tree* - TT) chamadas de *Binary-Ternary Tree* (BTT) ou de *Multi-Type Tree* (MTT), possibilitando a divisão de CUs retangulares, permitindo um melhor ajuste às características do conteúdo de cada imagem. Essas novas possibilidades de se dividir um bloco pode ser chamada também de

Árvore Quaternária Multi-tipo (em inglês *Quadtree with nested Multi-type Tree* - QTMT). Na Figura 7 é possível verificar a estrutura de particionamento de blocos QTMT, para uma CTU de tamanho 128 × 128 *pixels* e como são aplicadas as estruturas de particionamento QTMT, onde QT são representadas pelas linhas contínuas preta, BT representada pelas linhas pontilhas azuis e TT representada pelas linhas tracejadas laranja.

Figura 7 – Estrutura de particionamento de blocos QTMT.



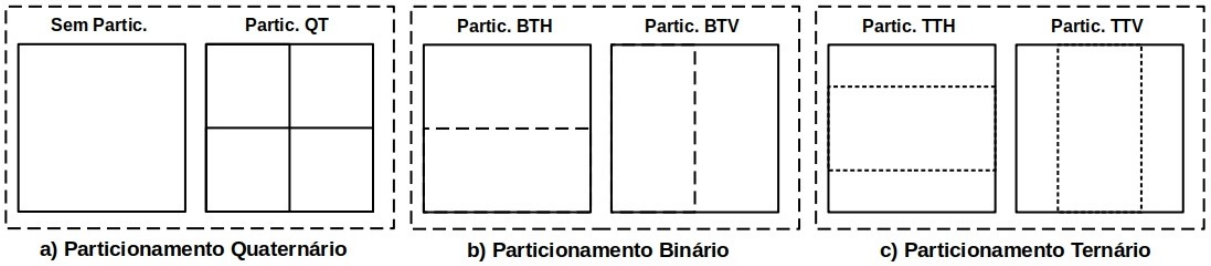
Fonte: (GONÇALVES, 2021).

No VVC cada CTU pode ser considerada uma CU ou pode ser dividida em múltiplas CUs por uma estrutura recursiva QT, seguida por uma estrutura recursiva MTT. Cada CTU é inicialmente particionada por uma estrutura QT, podendo não ser particionada se tornando um QT nó folha inteira, ou ser dividida em quatro QTs filha. Cada filha terá a forma quadrada e possuirá 1/4 do tamanho do nó pai, a Figura 8(a) ilustra o particionamento QT. Qualquer nó QT folha ele pode ser dividido recursivamente por MTT ou novamente por QT. Caso um nó folha QT seja dividido usando a estrutura MTT, não é mais permitida a divisão pela estrutura QT, que ficará desabilitada, podendo ser utilizada somente a estrutura MTT para os próximos particionamentos. Essa estratégia é adotada pelo VVC para simplificar o particionamento das CTUs (HUANG et al., 2021).

Seguindo a estrutura MTT, as CUs podem ser divididas em até quatro maneiras: *Binary Tree Horizontal* (BTH) e *Binary Tree* Vertical (BTV), dividem a CU em duas CUs filhas retangulares, como ilustrado na Figura 8(b) ou *Ternary Tree Horizontal* (TTH) e *Ternary Tree Vertical* (TTV) dividem a CU em três CUs filhas retangulares, ilustradas na Figura 8(c). Na estrutura BT cada CU filha possui o mesmo tamanho, com a metade

do tamanho da CU pai. Na estrutura TT a primeira, a segunda e a terceira CUs filhas possuem 25%, 50% e 25% do tamanho da CU pai, respectivamente.

Figura 8 – Formas de particionamento QT e MTT.



Fonte: Elaborado pelo autor.

Os níveis máximos de profundidade para as estruturas de QT e MTT são cinco e quatro, respectivamente, onde o tamanho mínimo de blocos com o particionamento utilizando a QT é de 8×8 *pixels*, mas utilizando a estrutura MTT os blocos podem alcançar até o tamanho 4×4 *pixels*. Portanto, a estrutura QTMT permite o particionamento de blocos em formatos quadráticos e retangulares com tamanhos de 4×4 até 128×128 *pixels*.

No VVC os conceitos de CU, *Prediction Unit* (PU) e *Transform Unit* (TU) são unificados, simplificando a estrutura de particionamento de blocos do VVC e reduzindo significativamente o *overhead* de sinalização para PUs e TUs sem sacrificar a eficiência de codificação (HUANG et al., 2021).

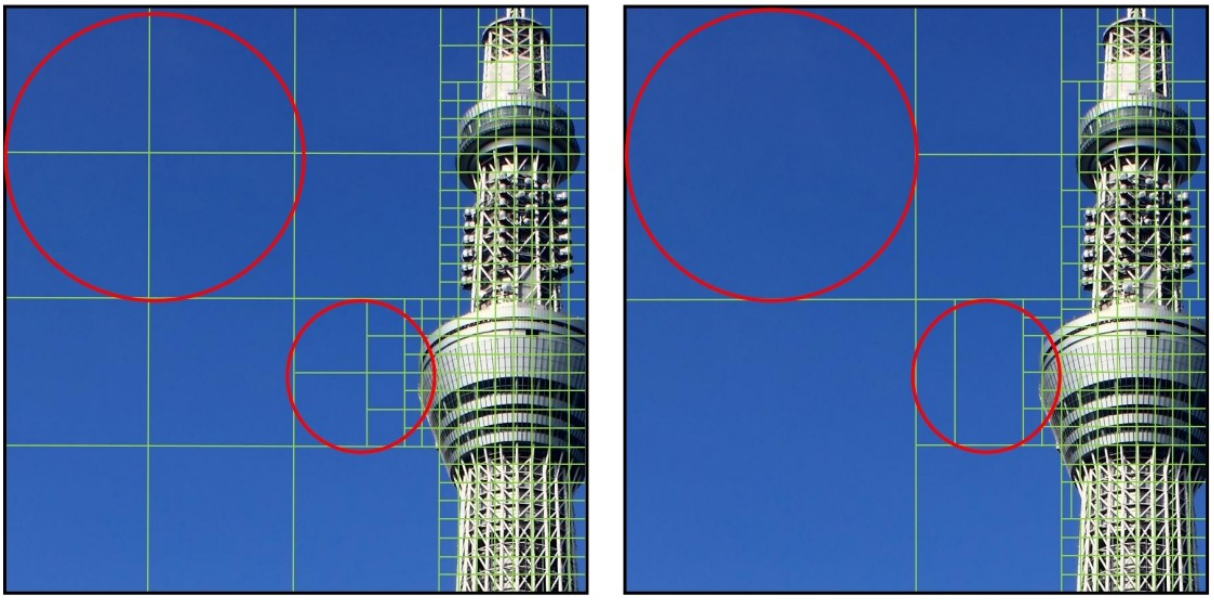
O particionamento de blocos do VVC é mais flexível que no HEVC e contribui significativamente para a eficiência e complexidade de codificação. Na Figura 9 é apresentada duas imagens, uma particionada com a estrutura QT do HEVC (Figura 9(a)) e a outra com a estrutura QTMT do VVC (Figura 9(b)). É possível perceber que no particionamento realizado pelo VVC há uma maior variedade de formas e tamanhos de CUs, como destacado nos círculos em vermelho, permitindo um melhor ajuste às características do conteúdo da imagem, com um particionamento mais fino nas regiões de borda e possibilidade de grandes CUs em região de maior similaridade.

2.3.2 Predição Inter-Quadro no VVC

VVC herda as principais ferramentas de predição inter-quadro do seu antecessor HEVC, realizando melhorias e inserindo novas ferramentas para obter melhores resultados em eficiência de codificação, reduzindo a taxa de *bits* e mantendo a qualidade do vídeo, sendo esta, uma das etapas que mais contribuem para esse objetivo. Uma introdução inicial à predição inter-quadros foi apresentada na seção 2.2.

Tanto HEVC como no VVC, utilizam quadros de referência para realizar a predição inter-quadros, como mencionado anteriormente os quadros de referência são quadros já

Figura 9 – Comparação entre o particionamento executado pelo HEVC e pelo VVC



Fonte: (GONÇALVES, 2021)

previamente codificados e reconstruídos. Para armazenar e gerenciar esses quadros, são criadas duas listas, chamadas de Lista de Quadros de Referência 0 (Lista 0) e Lista de Quadros de Referência 1 (Lista 1). Para armazenar os quadros de referência temporalmente anteriores ao quadro atual é utilizada a Lista 0 e os quadros de referência temporalmente posteriores ao quadro atual é utilizada a Lista 1. Para identificar cada quadro dentro de cada lista são utilizados índices de referência.

O processo de predição de compensação de movimento pode ser realizado de dois modos, unidirecional e bidirecional. Na predição unidirecional, são utilizado apenas os quadros de referência de uma das listas, podendo ser da Lista 0 ou a Lista 1. Já na predição bidirecional são utilizados dois quadros de referência para gerar a predição do bloco atual, um dos quadros de referência virá da Lista 0 e o outro virá da Lista 1, os dois blocos então são combinados para gerar o resultado final da predição.

O que define qual tipo de predição é utilizada é uma informação contida no cabeçalho do *slice*, que indica qual o tipo do *slice*, podendo ser *Slice* Preditivo (P *slice*) ou *Slice* bi-preditivo (B *slice*). Quando for P *slice*, somente predição unidirecional utilizando a Lista 0 é permitido, quando for B *slice*, é permitida a predição unidirecional utilizando a Lista 0 ou utilizando a Lista 1, além disso, na B *slice*, também é permitida a predição bidirecional, utilizando a Lista 0 e Lista 1 ao mesmo tempo.

A etapa de predição inter-quadro é composta de duas partes, a ME e a MC, como já descrito anteriormente. Durante a execução dessas etapas o codificador realiza a comparação dos blocos do quadro atual com os blocos dos quadros de referência que estão

armazenados nas listas (Lista 0 e/ou Lista 1, de acordo com a direção da predição), para cada bloco comparado é calculado um custo de taxa distorção (em inglês *Rate-Distortion* - RD).

Para o cálculo desse custo é utilizado o método de multiplicadores de Lagrange, que retorna a relação entre a qualidade objetiva do vídeo (distorção) e a quantidade de *bits* produzida (taxa de dados), ou seja, ele retorna um valor referente o equilíbrio entre qualidade da imagem e quantidade de dados gerada pela compressão (HOANG; LONG; VITTER, 1998). Assim o bloco que possuir o menor custo taxa distorção é o escolhido como o melhor bloco. A equação 2.3 apresenta a fórmula do método de multiplicadores de Lagrange, onde D representa o custo da distorção do bloco, λ representa o multiplicador Lagrange, e R representa o número estimado de bits para representar o bloco.

$$RD = D + \lambda * R \tag{2.3}$$

Ao final do processo da ME, são geradas as informações de movimento referente ao bloco escolhido, e que são utilizadas pela MC para realizar a reconstrução do quadro e então aplicar a codificação residual. Normalmente essas informações consistem na indicação da direção da predição (uni ou bidirecional), um ou dois vetores de movimento (dois MVs no caso da bidirecional), qual lista foi utilizada (Lista 0 ou Lista 1 no caso de ser unidirecional) e os índices de referência de cada lista associados aos MVs (para qual quadro de referência foi gerado o MV).

Em virtude da necessidade de enorme volume de comparações realizada pelos algoritmos de busca e as novas ferramentas incluídas e/ou melhoradas no VVC com o objetivo de melhorar a eficiência da codificação, a predição inter-quadros é considerada a etapa que apresenta a maior esforço computacional no VVC, representando em média 53% do tempo total de codificação (SIQUEIRA; CORREA; GRELLERT, 2020).

Com o objetivo de reduzir a taxa de bits gerada pelas informações de movimento, HEVC adota duas técnicas, sendo elas: modo avançado de predição do vetor de movimento (em inglês *Advanced Motion vector Prediction* - AMVP); e o modo *merge*. O AMVP realiza a predição do MV do corrente bloco baseado em informações de movimento de blocos vizinhos localizados espacialmente e temporalmente, previamente codificados. A diferença entre o MV os bloco vizinho e o MV encontrado pela ME é sinalizada como uma Diferença de Vetor de Movimento (em inglês *Motion Vector Difference*- MVD). Já o modo *merge* é verificado se algum bloco vizinho possui as mesmas informação de movimento do bloco atual, como vetores de movimento, índice do quadro de referência e a lista dos quadros de referência.

O modo *merge* reduz a taxa de *bits* de maneira mais eficiente que o AMVP por compartilhar todas as informações de movimento com os blocos vizinhos temporais e espaciais. Para um bloco que é definido como modo *merge*, somente *merge flag* e *merge index* devem ser sinalizados para indicar qual bloco vizinho faz o *merge* com a bloco atual.

O modo *merge* pode ainda ser executado como modo *skip*, não sendo necessário sinalizar nenhuma informação residual (KIM et al., 2018).

Os modos AMVP e *merge* são herdados e melhorados pelo VVC, proporcionando uma maior eficiência de codificação. Entre essas melhorias estão: permissão do uso de MVDs para o modo *merge*; realizar uma sinalização mais flexível do MVD para o modo AMVP, proporcionando uma maior compensação entre a precisão do movimento e os *bits* de movimento; estende o modo *merge* possibilitando a inclusão de mais dois tipos de blocos candidatos, *History-based Motion Vector Prediction* (HMVP) e o *Pairwise average Motion Vector Prediction* (PMVP) (CHIEN et al., 2021).

Além das melhorias, o padrão VVC implementa uma nova ferramenta para a predição inter-quadros chamada de Estimação de Movimento Afim (AME). Seu propósito é realizar uma codificação mais eficiente de movimentos não-translacionais, ou seja, movimentos irregulares como zoom, rotação e cisalhamento, que não são codificados eficientemente pelos modelos de movimentos translacionais.

Essa ferramenta será descrita com maiores detalhes na próxima seção.

2.3.3 Estimação de Movimento *Affine*

Nos padrões de codificação de vídeo anteriores ao VVC, a estimação de movimento utilizava apenas modelos de predição de movimentos translacionais para detectar todos os tipos de movimentos. Porém estudos anteriores como (VEERAVALLI et al., 2005; BRADSHAW; KINGSBURY, 1997) demonstravam que técnicas de predição de movimentos não-translacionais são capazes de codificar movimentos de rotação, zoom e cisalhamento com maior precisão, proporcionando uma predição mais eficiente do que as técnicas de movimentos translacionais.

Para solucionar esse problemas, modelos mais sofisticados começaram a ser pesquisados para trabalhar com movimentos não-translacionais. O trabalho do autor (ZHANG et al., 2019) descreve o percurso realizado pelo modelo de estimação de movimento *affine* (AME), desde as pesquisas iniciais até sua implementação em um padrão de codificação de vídeo. Inicialmente foram propostos modelos de algoritmo de casamento de blocos deformáveis, onde um MV para qualquer posição dentro do bloco pode ser calculado por um modelo polinomial envolvendo vários MV em Pontos de Controle (em inglês *Control Points* - CP). No geral, algoritmos de casamento de blocos deformáveis podem representar uma grande gama de modelos de movimentos complexos, como *affine*, bilinear e projetivo. Pesquisas realizadas entre esses modelos mostraram que o modelo que permite representar a maioria dos movimentos não-translacionais presentes no mundo real é o *affine*. A partir de então, a AME, tem recebido atenção nas últimas três décadas.

Um AME global é uma forma direta de introduzir a AME em um codificador de vídeo. Ele assume que todos os movimentos em uma imagem seguem um consistente modelo *affine*. Ele funciona bem quando a imagem como um todo está girando ou ampliando, porém,

não apresenta um bom desempenho se em diferentes regiões da imagem são aplicados diferentes tipos de modelos de movimentos, por exemplo: em uma parte da imagem está rotacionando e outra ampliando. Para contornar esse problema um modelo de AME local foi desenvolvido, proporcionando uma maior flexibilidade para permitir diferentes tipos de modelos de movimento não-translacionais dentro de uma mesma imagem, para isso a imagem é dividida em malhas, que segmentam regiões com diferentes modelos de movimento *affine*.

Ainda assim, executar a ME em um esquema AME continua sendo muito difícil, uma vez que a busca pelo bloco mais similar baseada em ME possui uma extrema complexidade computacional para derivar os MVs no modelo *affine*. Então foi proposto uma estratégia de ME rápida baseado em gradiente iterativo para o modelo *affine*, largamente usada e melhorada em posteriores trabalhos de pesquisa.

Apesar de AME ter sido um tópico popular em pesquisas por um bom tempo, ela não foi considerada uma ferramenta de codificação de vídeo viável até os últimos anos. Para isso, duas barreiras deveriam ser vencidas, a primeira é referente a complexidade computacional imposta pela AME, tanto no codificador como no decodificador, tornando impossível sua aplicação prática. A segunda diz respeito à dificuldade de desenvolver um *framework* eficiente e elegante, que possa inserir a AME em um codificador baseado em blocos seguindo o conceito do padrão de codificador de vídeo híbrido (ZHANG et al., 2019).

Essa situação se altera após a finalização do padrão HEVC. Com a dificuldade cada vez maior de conseguir mais eficiência de codificação utilizando modelos de movimentos translacionais, os modelos AME se tornaram cada vez mais atrativos.

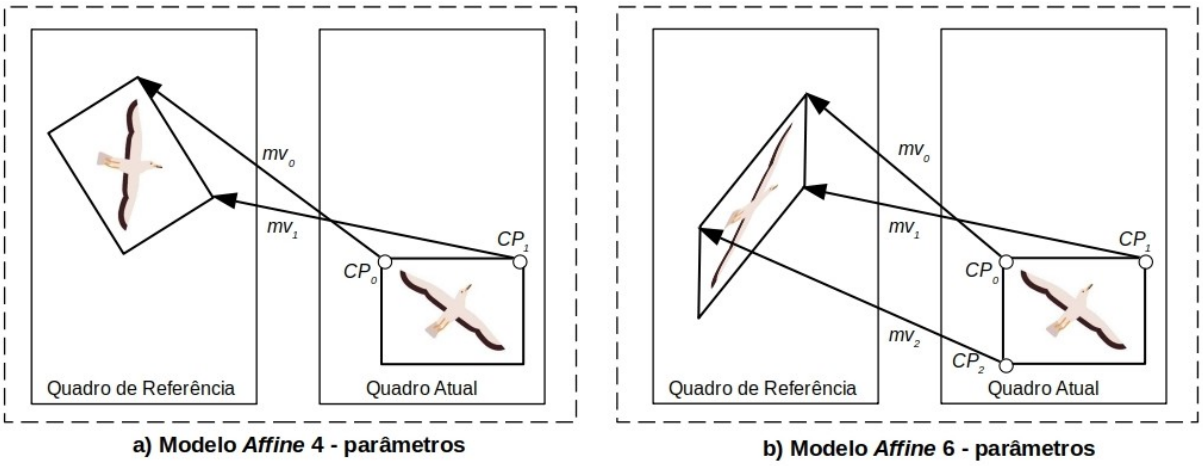
Estudos recentes demonstraram que a AME consegue superar as barreiras mencionadas anteriormente. Como resultado desses estudos, foi proposto o modelo *affine* com 6-parâmetros representados por MVs para três CPs baseado no HEVC, e ainda foi incorporado a AME como candidato *merge* especial também no HEVC. Como simplificação ao modelo com 6-parâmetros, um modelo *affine* com 4-parâmetros foi aplicado. Modo *affine* inter e modo *affine merge* foram desenvolvidos como algoritmos de predição de MV. O método de ME rápida baseado em gradiente iterativo foi revisado e aprimorado desenvolvendo uma baixa complexidade do lado do codificador. Além disso, a granularidade no modelo *affine* pode ser ampliada para o nível de sub-bloco com tamanho 4×4 *pixels*, restringindo a complexidade do decodificador.

Após as pesquisas descritas acima, foi encontrado um equilíbrio interessante entre a complexidade e eficiência de codificação, então um *framework* em acordo com as normas de codificação foi proposto para o JVET e implementado no JEM. Nesse momento a nova ferramenta de predição inter-quadros AME, foi introduzida no Padrão de Codificação VVC (ZHANG; MAO, 2019).

Para gerar os MV para o AME, o VVC pode escolher entre dois modelos de

movimento *affine* de acordo com o número de pontos de controle. Um deles é o modelo *affine* de 4-parâmetros com dois CPs, esse modelo é descrito por MVs de dois pontos de controle localizados nos cantos superior-esquerdo (CP_0) e superior-direito (CP_1) do bloco atual, como mostrado na Figura 10(a), onde o vetor de movimento mv_0 é associado ao CP_0 e o mv_1 associado ao CP_1. O outro é o modelo de 6-parâmetros com três CPs, esse modelo é descrito por MVs de três pontos de controle localizados nos cantos superior-esquerdo (CP_0), superior-direito (CP_1) e inferior-esquerdo (CP_2) do bloco atual, como mostrado na Figura10(b), onde o vetor de movimento mv_0 é associado ao CP_0, mv_1 associado ao CP_1 e o mv_2 associado ao CP_2 (BROSS et al., 2021a).

Figura 10 – Modelos de predição *affine* 4 e 6-Parâmetros



Fonte: Elaborado pelo autor.

Dessa forma, os vetores de movimento para cada amostra do bloco, nas posições (x,y) são calculados a partir da equação 2.4 para o modelo *affine* com 4-parâmetros, onde w representa a largura do bloco atual, (mv_{0x}, mv_{0y}) é o MV associado ao CP_0 e (mv_{1x}, mv_{1y}) é o MV associado ao CP_1, e da equação 2.5 para o modelo *affine* de 6-parâmetros, onde w representa a largura e h a altura do bloco atual, (mv_{0x}, mv_{0y}) é o MV associado ao CP_0, (mv_{1x}, mv_{1y}) é o MV associado ao CP_1 e (mv_{2x}, mv_{2y}) é o MV associado ao CP_2.

$$\begin{cases} mv_x = \frac{mv_{1x}-mv_{0x}}{w}x + \frac{mv_{1y}-mv_{0y}}{w}y + mv_{0x} \\ mv_y = \frac{mv_{1y}-mv_{0y}}{w}x + \frac{mv_{1y}-mv_{0x}}{w}y + mv_{0y} \end{cases} \tag{2.4}$$

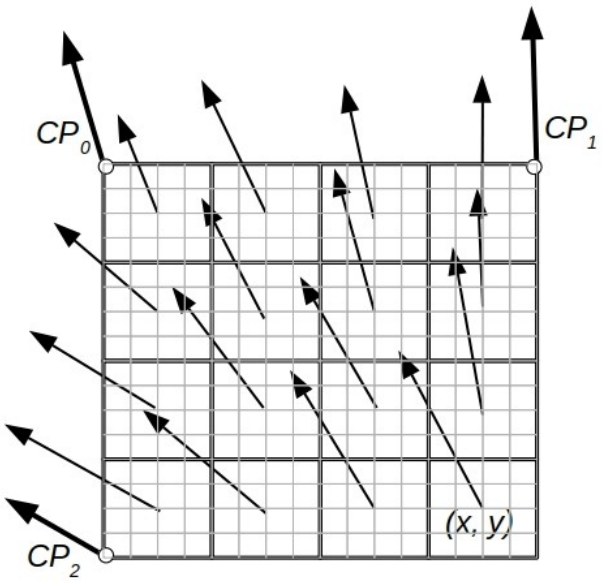
$$\begin{cases} mv_x = \frac{mv_{1x}-mv_{0x}}{w}x + \frac{mv_{2x}-mv_{0x}}{h}y + mv_{0x} \\ mv_y = \frac{mv_{1y}-mv_{0y}}{w}x + \frac{mv_{2y}-mv_{0y}}{h}y + mv_{0y} \end{cases} \tag{2.5}$$

Os modelos *affine* podem representar uma grande variedade de movimentos complexos, e de acordo com a complexidade desses movimentos, se aplica o modelo *affine* de 4 ou 6-parâmetros, por exemplo, zoom e rotação podem ser representados por dois CPs

como mostrado na Figura 10(a), já cisalhamento é necessário 3 CPs como mostrado na Figura10(b).

Com o objetivo de reduzir a complexidade computacional da AME, VVC aplica a AME baseada em sub-blocos, onde os blocos são divididos em sub-blocos de tamanho 4×4 *pixels*, o MV de cada sub-bloco é derivados do centro de cada sub-bloco de acordo com as equação 2.4 ou 2.5, em vez de ser calculado para cada amostra do sub-bloco. Após o MV é arredondado para precisão fracionária de 1/16 e então um filtro de interpolação *six-tap* é aplicado. O MV derivado é compartilhado com todos as amostras/*pixels* dentro do sub-bloco, como mostrado na Figura 11 (LIU et al., 2019).

Figura 11 – Sub-bloco 4×4 de uma CU 16×16 para predição *affine*.



Fonte: Elaborado pelo autor.

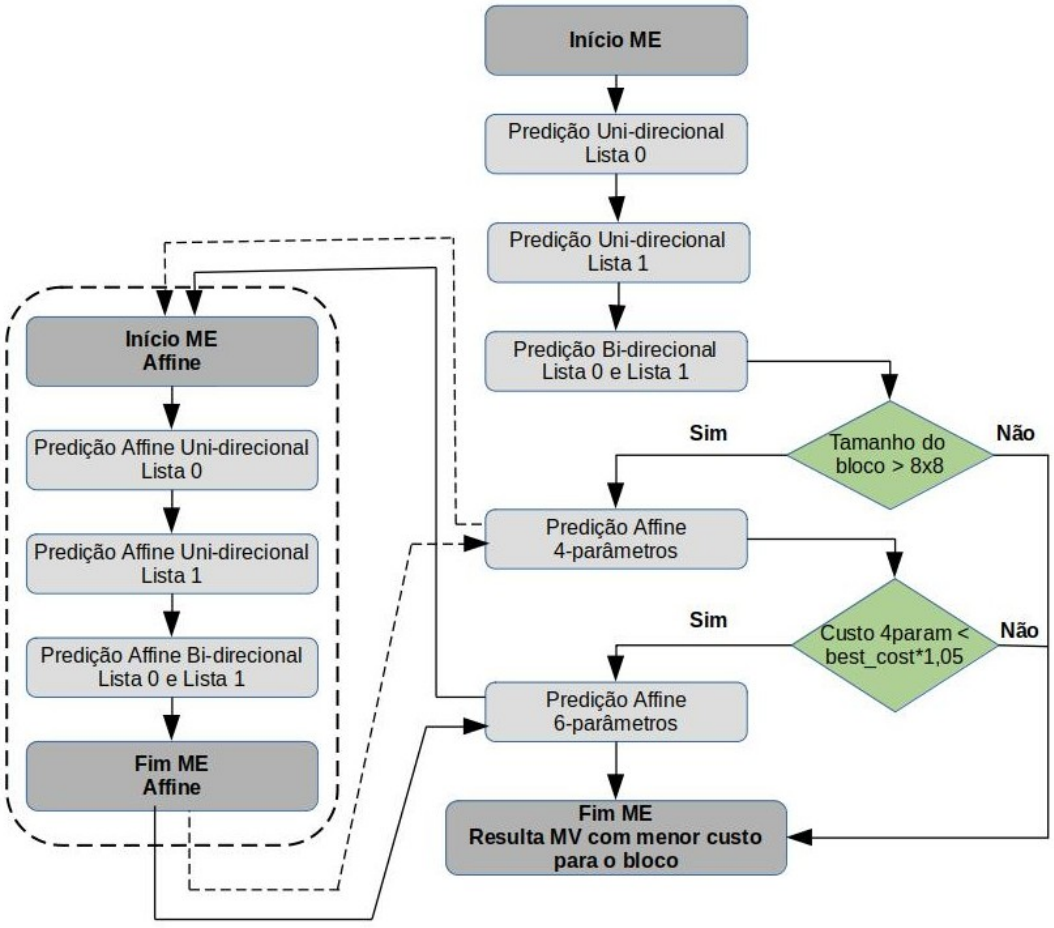
Assim como nos modos *merge* e AMVP translacionais, o VVC também implementa esses modos no AME, melhorando a predição e a codificação dos parâmetros de movimento *affine* (BROSS et al., 2021b). No modo *affine merge* os dados de movimento dos pontos de controle (em inglês *Control Pointer Motion Vector* - CPMV) do bloco atual são derivados das informações de movimento de blocos espacialmente vizinhos. A lista de candidatos *merge* pode ter até cinco candidatos, e um único índice é sinalizado no *bitstream*, indicando o bloco vizinho que irá realizar o *merge* com o bloco atual. Podem ser aplicados em blocos com tamanhos maior ou igual a 8×8 *pixels*.

No modo *affine* AMVP, uma *affine flag* é sinalizada no *bitstream* indicando se o *affine* AMVP é utilizado e outra *flag* é sinalizada indicando se é o modelo de 4 ou 6-parâmetros. Ainda é sinalizado no *bitstream* também a diferença entre o CPMVs do bloco atual e os seus CPMVs preditos (do bloco escolhido pelo modo AMVP). A lista de candidatos do modo afine AMVP pode ter até 2 candidatos e é construída com blocos

espacialmente vizinhos. Esse modo suporta bloco de tamanho maior ou igual a 16×16 (YANG et al., 2021).

A Figura 12 apresenta um esquema com o fluxo de execução da estimação de movimento executada no VVC. Inicialmente a CU é processada pela estimação de movimento convencional (CME), passando pelas etapas Unidirecional e Bidirecional. Na sequência a CU é avaliada para verificar se sua largura e altura são maiores que 8, sendo verdadeira a resposta da condição, então a estimação de movimento *affine* (AME) é executada, caso contrário ela sera totalmente ignorada. Na AME existem os modelos com 4 e 6-parâmetros, cada um deles deve executar as etapas Unidirecional e Bidirecional da própria *affine*. Porém o modelo *affine* de 6-parâmetros só é executado caso o custo do modelo *affine* com 4-Parâmetros seja menor que o melhor custo da CME×1,05.

Figura 12 – Esquema de execução da ME no VVC.



Fonte: Elaborado pelo Autor.

Como pode ser analisado na Figura 12, para cada uma das etapas da AME (com 4 e 6 - parâmetros) a predição Unidirecional e Bidirecional *affine* é executada. Além disso, levando em consideração que para cada Ponto de Operação deve ser atribuído um vetor de movimento e que posteriormente será usado para derivar o vetor de movimento

do centro de cada sub-bloco 4×4, todo esse processo torna a AME muito complexa computacionalmente. Este processo pode representar, como mencionado anteriormente, mais de 50% de todo o tempo da ME.

Uma maneira de reduzir o custo computacional do VVC é evitar que determinados tamanhos de CUs sejam processados pela etapa da AME. A AME seria ignorada quando provavelmente a *affine* não fosse escolhida como o melhor modo de predição para a CU. Para realizar essa classificação técnicas de Aprendizado de Máquina pode ser utilizado.

No próximo capítulo são apresentados os conceitos de Aprendizado de Máquina, Árvores de Decisão e Florestas Aleatórias utilizados para o desenvolvimento da proposta baseada em otimização de software.

3 APRENDIZADO DE MÁQUINA

Aprendizado de Máquina (em inglês *Machine Learning* - ML) pertence a um dos ramos da Inteligência Artificial. São algoritmos computacionais capazes de aprimorar seu desempenho automaticamente através da experiência obtida com o acesso à informações externas. Eles melhoram suas previsões ou decisões sem explicitamente serem programados para isso. Segundo o criador do termo Aprendizado de Máquina (SAMUEL, 1959), Aprendizado de Máquina é o campo de estudo que dá aos computadores a habilidade de aprender sem terem sido programados para tal finalidade.

Segundo (CHINNAMGARI, 2019) existem vários problemas do mundo real que podem ser resolvidos utilizando técnicas de Aprendizado de Máquina. ML possui grupos de algoritmos específicos para resolver determinados tipos de problemas. Os grupos mais comuns são: Aprendizado Supervisionado, Aprendizado Não Supervisionado e Aprendizado por Esforço.

Aprendizado supervisionado: esse método é baseado em observações anteriores, utilizado quando já se possui um conjunto de instâncias previamente extraídas.

Cada instância pode ser representada por um vetor possuindo os valores das variáveis. As variáveis podem ser divididas em duas categorias: **(i)** variáveis de entrada (também chamadas de *features*), que representam os valores utilizados para a tomada de decisão, e **(ii)** variável de saída (também chama de classe ou *target*) que representa a saída/resultado de cada instância. Variáveis de saída são utilizadas somente nas etapas de treinamento e testes do modelo.

Nesse método de aprendizado as instâncias de entrada devem ser rotuladas, ou seja, para cada conjunto de variáveis de entrada uma informação de saída deve estar presente, por isso o nome de aprendizado supervisionado. Com as instâncias rotuladas, o algoritmo irá "aprender" através da identificação dos padrões que ocorrem entre as variáveis de entrada e as de saída. Essa etapa de identificação de padrões é chamada de treinamento.

Como resultado do processo de treinamento é gerado um algoritmo "treinado" chamado de modelo. Para validar o modelo, um conjunto de instâncias de teste, diferente dos utilizados para o treinamento, é submetido ao modelo para mensurar o grau de precisão das respostas. O modelo validado deverá ser capaz de generalizar para novas instâncias de entrada, agora sem os valores de saída.

O aprendizado supervisionado ainda pode ser dividido em duas categorias, classificação e regressão (BREIMAN, 2001). **(i)** modelos de classificação são indicados quando as respostas de saída podem ser categorizadas por classes (valores discretos), por exemplo, masculino e feminino. Modelos multi classe também são possíveis, permitindo que mais de duas classes sejam previstas pelo modelo; **(ii)** o modelo de regressão é indicado quando as informações de saída são representadas por valores numéricos contínuos, não sendo possível a identificação por classe, como exemplo podemos citar um modelo para prever o

preço de veículos, que de acordo com as características do veículo, o modelo deve sugerir o seu preço.

Aprendizado não supervisionado: Esse método de aprendizado é indicado para modelos que não possuem instâncias rotuladas ou classificadas, ou seja, o modelo receberá somente variáveis de entrada. O algoritmo é treinado através da identificação de padrões entre as instâncias de entrada, buscando classificá-las de acordo com suas semelhanças e diferenças.

Aprendizado por esforço: Como no aprendizado não supervisionada, somente variáveis de entrada são utilizadas. O algoritmo irá ser treinado através de recompensas ou punições de acordo com seu acertos e erros. Assim o algoritmo irá ser recompensado positivamente a cada acerto e negativamente a cada erro, consequentemente melhorando seu desempenho a cada nova simulação.

Os modelos de classificação podem ser utilizado em diferentes áreas como medicina, industria, marketing e também em codificadores de vídeo. Na literatura vários trabalhos foram encontrados utilizando técnicas de ML com o objetivo de reduzir o esforço computacional do VVC. Em (AMESTOY et al., 2020; ZHANG et al., 2020; CHEN et al., 2022; LI; ZHANG; YANG, 2022; HE et al., 2021) o método de RFC foi aplicado principalmente na etapa de particionamento de blocos QTMT, já os trabalhos (SALDANHA et al., 2022a) e (SALDANHA et al., 2022b) utilizam outras técnicas de ML para reduzir o esforço computacional do VVC como *Light Gradient Boosting Machine* (LGBM) e Árvores de decisão respectivamente.

Os trabalhos listados acima apresentam bons resultados utilizando técnicas de classificação com modelos de aprendizado supervisionado, resultando em um bom *tradeoff* entre a processamento dos modelos e o desempenho dos classificadores. Baseado nos trabalhos analisados, neste trabalho será utilizado o método de Classificação Florestas Aleatórias (RFC) para realizar a classificação das CUs, definindo se elas são ou não ignoradas pela etapa da AME.

3.1 Árvores de Decisão

Árvores de decisão (em inglês *Decision Tree* - DT) é um dos mais populares modelos de classificação, no qual os modelos seguem o conceito de estrutura em árvore. Esse modelo é um mapa ou grafo utilizado para avaliar o curso de ação resultando em uma predição probabilística (PAPAGEORGIOU; STYLIOS; GROUMPOS, 2006; MAIMON; ROKACH, 2010).

As DTs utilizam a abordagem "dividir para conquistar" onde o conjunto de instâncias de treinamento é dividido recursivamente em subconjuntos, chamados de nós ou sub-árvores, até obter o valor esperado ou atingir alguma limitação.

O processo de recursão é encerrado em um desses casos: **(i)** o número de variáveis é menor do que o limite determinado; **(ii)** as variáveis do nó atual pertencem todas a

mesma classe; **(iii)** os valores das variáveis do nó atual estão vazios; **(iv)** a profundidade da árvore atinge o valor pré definido.

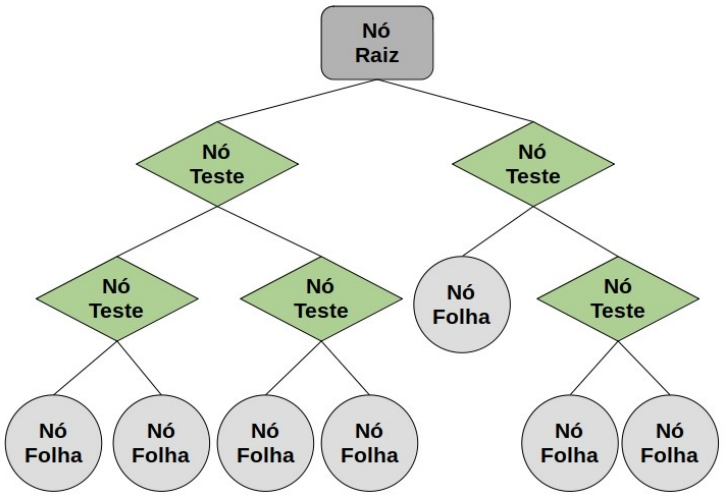
DT é formada por três tipos de nós: nó raiz, intermediários ou teste e folha ou terminal.

- O nó raiz é o primeiro nó da DT, possuirá apenas arestas de saída e indica o início da classificação de cada instância;
- Nós de teste ou intermediário possuem arestas de entrada e saída. Dividem o conjunto de instâncias, em dois ou mais subconjuntos. Essa divisão ocorre após algoritmos processarem os valores das variáveis (relacionando variáveis de entradas com as variáveis de saída) de cada instância determinando a melhor forma de separação;
- Nós folha ou terminal, possuem apenas arestas de entrada. Representados pelos nós mais profundos da árvore, contém o valor da variável classe, indicando qual o resultado da árvore de decisão.

Dessa maneira, o processo de classificação de cada instância inicia no nó raiz e finaliza em um nó folha. O caminho percorrido dentro da árvore é definido pelas respostas obtidas em cada nó de teste.

A Figura 13 mostra a representação genérica e uma Árvore de Decisão, onde o nó raiz é representado por um retângulo, os nós de teste são representados por losangos e os nós folha representados por círculos.

Figura 13 – Esquema genérico de um Árvore de Decisão.



Fonte: Elaborado pelo Autor.

Durante a construção das árvores alguns métodos podem ser utilizado para determinar a melhor forma de divisões entre os nós. Na maioria dos casos são utilizadas

funções discretas, onde o critério de divisão é definido com base no valor de uma única variável.

Existem muitos métodos para medir o grau de impureza (não-homogeneidade) de um conjunto de instâncias. Os dois mais utilizados pelas DTs são Índice Gini e a Entropia. 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; MAIMON; ROKACH, 2010).

Índice Gini é uma métrica que mede a frequência que uma variável aleatória do conjunto de instâncias vai ser rotulada erroneamente. Os valores possíveis para o Índice Gini estão entre 0 e 0,5. Quanto mais próximo a zero for o valor do índice melhor é, pois mais homogêneo (pura) é a variável, indicando que o nó pode ser dividido. Por exemplo, se para uma determinada variável o Índice Gini for zero, indica que a variável contribui para a escolha de uma determinada classe com 100% de probabilidade de acerto. Por outro lado, se o valor for 0,5 indica que a variável terá a mesma probabilidade de acerto entre todas as classes (SUTHAHARAN, 2016). O cálculo do Índice Gini é representado pela equação 3.1, onde p_j representa a probabilidade da classe j.

$$GiniIndex = 1 - \sum_j p_j^2 \tag{3.1}$$

Entropia, é uma métrica que indica a desordem entre a variáveis de entrada em relação a classe. Similar ao Índice Gini a melhor divisão é dada pela variável com menor entropia. Os valores possíveis então entre 0 e 1. Quando mais próximo de zero menor a entropia e mais homogêneos são as variáveis, indicando que a variável tende a identificar melhor uma determinada classe. Caso o valor seja 1, indica que o grau de entropia é igual entre todas as classes (SUTHAHARAN, 2016).

O cálculo da entropia é dado pela equação 3.2, onde a p_j representa a probabilidade de cada classe.

$$Entropia = -\sum_j p_j \times log_2 \times p_j \tag{3.2}$$

Uma das diferenças entre Índice Gini e Entropia, podemos citar que a Entropia pode ser mais complexa computacionalmente do que o Índice Gini devido a utilização de cálculos logarítmicos, por outro lado a Entropia pode indicar divisões mais precisa durante a criação da árvore.

As árvores de decisão possuem uma alta interpretabilidade devido a facilidade de ser representada graficamente e analisar quais critérios foram utilizado para o processo de classificação. No entanto, apresentam uma alta variância, sendo sensíveis a mudanças nos dados de treino. Pequenas mudanças no dados podem gerar resultados muito diferentes. Além disso o modelo é muito suscetível ao *overfitting* (sobre ajuste), adaptando-se muito bem aos dados de treino mas possuindo pouca eficácia na classificação de novas instâncias (HARRINGTON, 2012).

3.2 Florestas Aleatórias

O método Florestas Aleatórias (em inglês *Random Forest* - RF) foi proposto por (BREIMAN, 2001) extraindo as vantagens das árvores de decisões e contornando suas limitações.

RF é um classificador *ensemble*, que utiliza o resultado de diferentes classificadores para obter o resultado final (PARMAR; KATARIYA; PATEL, 2018).

O modelo RF é formado por um conjunto de múltiplas DTs. Operações de aleatoriedade são introduzidas no processo de criação da RF, incluindo a seleção de subconjuntos de instâncias e subconjuntos de variáveis, garantindo a independência de cada DT, melhorando a precisão da classificação e a habilidade de generalização. As operações introduzidas durante o processo de criação das RF permitem resolver o problema de instabilidade de uma única DT evitando o *overfitting* do modelo, além disso a independência de cada DT dentro da RF possibilita a paralelização da codificação das DTs durante o processo de treinamento e classificação.

O resultado final da RF é obtido através de um algoritmo de votação que calcula o número de votos resultante de cada DT que faz parte da RF, então a classe que obtiver o maior número de votos é selecionada. O algoritmo de votação pode utilizar a contagem ou a média dos votos para calcular o resultado final.

A Figura 14 mostra uma representação genérica de uma RF, onde as instâncias de entrada são classificadas por diferentes DTs, e então o algoritmo de votação recebe o resultado de cada DT e retorna o resultado final da RF.

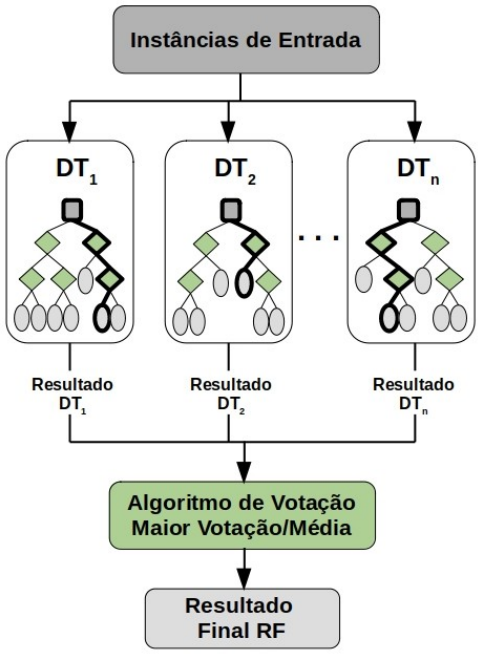
Abaixo são elencadas algumas vantagens de se usar RF do ponto de vista computacional (CUTLER; CUTLER; STEVENS, 2012):

- Trabalhar naturalmente com classificação e regressão (multi-classe);
- Treinamento e classificação relativamente rápidos;
- Com dois ou três parâmetros de ajuste, já se consegue bons resultados;
- Ter uma estimativa de erro de generalização embutida;
- Pode ser usado diretamente para problemas de alta dimensionalidade;
- Fácil implementação em paralelo;

Do ponto de vista estatístico, as RF podem ser mais indicada pelos recursos adicionais que proporcionam, como:

- Medida de importância das variáveis;
- Ponderação diferente entre classes;
- Imputação de dados faltantes;

Figura 14 – Esquema genérico de uma Floresta Aleatória.



Fonte: Elaborado pelo Autor.

- Fácil visualização e compreensão das DTs;
- Detecção de *outliers*;

3.2.1 Métricas de Validação dos modelos

Para a validação dos modelos de ML como o RFC, são utilizadas métricas que demostram o desempenho dos modelos durante o treinamento, ou seja, através delas é possível analisar se o modelo está alcançando a precisão desejada. Algumas dessas métricas são a *feature importance*, acurácia, *recall*, precisão e *F1-score*.

- ***Feature importance***: essa métrica atribui para cada *feature* um determinado valor que varia entre 0 e 1 (a soma dos valores de todas as *features* deve resultar em 1). Quanto mais próximo a 1 for o valor atribuído a *feature* maior é a sua contribuição para o modelo, ou seja, mais importante para o modelo ela é.
- **Acurácia**: indica uma desempenho geral do modelo. Dentre todas as classificações, quantas o modelo classificou corretamente, quanto mais próximo a 1 for o resultado da métrica melhor;
- ***Recall***: entre todas as situações de classe 1 (resultado positivo) como valor esperado, quantas estão corretas, quanto mais próximo a 1 for o resultado da métrica melhor;

- **Precisão**: entre todas as classificações de classe 1 (resultado positivo) que o modelo fez, quantas estão corretas, quanto mais próximo a 1 for o resultado da métrica melhor;
- ***F1-Score***: obtida pela média harmônica entre precisão e *recall*, quanto mais próximo a 1 for o resultado da métrica melhor;

4 TRABALHOS RELACIONADOS

A ME é uma das técnicas mais importantes para a compressão de dados de vídeos, pois ela explora as redundâncias entre os quadros que formam o vídeo. Nos padrões de codificação de vídeo anteriores ao VVC, a estimação de movimento utilizava apenas técnicas de predição de movimentos translacionais para detectar todos os tipos de movimentos. Porém estudos anteriores como (VEERAVALLI et al., 2005; BRADSHAW; KINGSBURY, 1997) demonstravam que técnicas de predição de movimentos não-translacionais, capazes de codificar movimentos de rotação, zoom e cisalhamento, apresentaram uma maior precisão na predição desses movimentos, proporcionando maior eficiência na compressão dos dados.

Com a evolução da capacidade de processamento do hardware, técnicas de processamento paralelo, compartilhamento de memória, entre outras, permitiram que essas técnicas de detecção de movimentos não-translacionais, que possuem um alto custo computacional, fossem inseridas no novo padrão de codificação VVC, sendo chamada de AME.

Apesar das técnicas de estimação de movimento não-translacional para codificadores de vídeo estarem sendo estudadas a pelo menos três décadas, sua implementação em um codificador de vídeo ocorreu recentemente no VVC (ZHANG et al., 2019), devido a isso ainda não se encontra na literatura muitos trabalhos que focam exclusivamente na estimação de movimentos *affine*. Abaixo são listados três trabalhos focados na redução da complexidade da AME. Os dois primeiros são baseados em análises estatísticas e o terceiro aplica técnicas de Aprendizado de Máquina.

O trabalho desenvolvido por (PARK; KANG, 2019), propõe um método para reduzir o esforço computacional da ME no VVC. Ele consiste em definir como a *Coding Unit* (CU) atual é processada baseado nas informações da CU pai já previamente codificada, para isso um esquema de duas etapas é executado. Na primeira etapa é identificado o melhor modo de predição escolhido para a CU pai, caso o melhor modo de predição da CU pai for *Skip* os demais sub-blocos filhos não são testados pela AME. A segunda etapa é executada caso a primeira etapa não seja satisfeita, nesse caso, é identificada qual a lista de referência foi escolhida pela predição na ME convencional, caso seja a Lista 0, a AME irá ser executada utilizando somente o conjunto de *frames* da L0, reduzindo o número de *frames* testado. Os resultados do experimento apontaram que o tempo total da AME teve uma redução média de 4% na segunda etapa, já o resultado da primeira etapa alcançou uma média de redução maior, chegando a 31%, comparados com o teste de referência. Quando analisado o resultado completo, o esquema proposto apresentou uma redução no tempo de em média 37% no tempo total da AME, e 5% no tempo total médio de codificação, com uma perda de eficiência de codificação de 0,1%.

O trabalho de (JUNG; JUN, 2021) propõe um método para reduzir o esforço computacional do VVC focando na AME. A proposta é desenvolvida em duas etapas. A primeira é etapa verifica se a CU pai foi codificada com o modo *affine* e também se ela possui coeficiente diferente de zero, caso as duas condições sejam satisfeitas, um segundo

teste irá ocorrer, verificando se a distância entre o quadro atual e o quadro de referência for diferente de 4, e também se a CU atual possui como melhor modo de predição o *Merge Affine*. Caso ao menos uma das condições acima não sejam satisfeitas a AME inteira é pula pela CU atual, caso contrário a segunda etapa é executada. Na segunda etapa é verificado se o o custo da AME 4-parâmetros da CU atual é menor que o custo da AME 6-parâmetros da CU pai, caso a condição não seja satisfeita a AME 6-parâmetros da CU atual é pulada. Os resultados alcançados mostram uma redução de 33% na complexidade de codificação na etapa ME *Affine*, 5% no tempo total de codificação com 0,04% de perda na eficiência de codificação (BD-BR).

A proposta apresentada por (DUARTE et al., 2022), traz como objetivo reduzir o esforço computacional da etapa ME, para isso apresenta um esquema baseado em Aprendizado de Máquina chamado *Learning-based Affine Prediction* (LEAP). O esquema define seis modelos de Florestas Aleatórias que tem por objetivo decidir encerrar a etapa de predição *Affine* em dois estágios, um anterior a etapa de predição *Affine* de 4-parâmetros e outro anterior a etapa de predição *Affine* de 6-parâmetros. Os resultados do experimento mostraram que o LEAP foi capaz de reduzir o tempo médio de codificação total em 8,49% e o tempo de codificação da etapa de predição *affine* foi reduzido em 46,94%, quando comparados com os testes de referência. Quanto à eficiência de codificação, os resultados apresentaram um acréscimo de 0,18%.

Abaixo são listados alguns trabalhos contendo propostas de redução do esforço computacional do VVC, através da utilização de técnicas de Aprendizado de Máquina.

O trabalho de (AMESTOY et al., 2020) apresenta uma proposta para reduzir o esforço computacional do VVC, aplicando a técnica de classificação Florestas Aleatórias na etapa de particionamento de blocos *Quad Tree Binary Tree* (QTBT). QTBT foi modelado em três categorias distintas de divisão binária, em cada categoria, para cada tamanho de CU, foi criado e treinado um modelo de classificação de Florestas Aleatórias. O objetivo é pular a custosa exploração dos modos de particionamento classificados pelo modelo de Florestas Aleatórias como desnecessários. O experimento apresenta uma redução na complexidade do codificador variando entre 30% e 70%, com uma perda de qualidade de imagem variando entre 0,7% e 3% quando implementado no JEM-7.0. A solução foi implementa também no VTM-5.0, incluindo o novo modo de particionamento ternário, esse nova implementação obteve uma redução no esforço computacional variando de 25% a 61%, com uma perda de qualidade de imagem variando entre 0,4% e 2,2%.

Em (ZHANG et al., 2020) é proposto a redução da complexidade e aumento na economia de tempo do codificador VVC, através de um algoritmo para acelerar o particionamento de blocos e também os modos de predição intra. Para acelerar o particionamento de blocos é utilizado o classificador Florestas Aleatórias, determinando se o particionamento da CU atual deve ser realizado ou não. Para acelerar os modos de predição intra, características das regiões de textura são utilizadas para determinar se

o modo de predição intra deve ser executado para a CU atual ou não. Os resultados do trabalho apresentam uma economia de 54,91% o tempo total de codificação, com uma perda de qualidade de imagem de 0,93%.

O trabalho realizado por (CHEN et al., 2022), propõe a redução do esforço computacional do codificar VVC através de um rápido particionamento de blocos para predição intra, utilizando classificador Florestas Aleatórias. Para a criação dos modelos foi explorada as características do sistema visual humano e aplicadas ao modelo de Aprendizado de Máquina. O modelo irá determinar se a CU atual deverá ser particionada ou não. Os resultados apresentam uma economia de 40,57% no tempo total de codificação com uma perda de qualidade de imagem de 1,14%.

A proposta de (LI; ZHANG; YANG, 2022) apresenta um algoritmo de rápido particionamento de blocos para a predição inter baseado na técnica de Florestas Aleatórias. Utilizando informações de domínio de tempo e de complexidade de texturas extraídas da CU atual, os modelo de Florestas Aleatórias foi treinados para determinar se o particionamento da CU deve ocorrer ou não. Os resultados demonstram uma redução média no tempo total de codificação de 15,27%, com uma redução na qualidade da imagem de 0,07%.

(HE et al., 2021) propõe um algoritmo baseado em Florestas Aleatórias para reduzir a esforço computacional do particionamento de blocos QTMT. As CU são divididas em três categorias, simples, intermediárias e complexas. Para CUs simples e complexas, um modelo de classificação Florestas Aleatórias é treinado para predizer o melhor modo de particionamento. Para CUs intermediárias outro classificador de Florestas Aleatórias é treinado para decidir se o processo de particionamento é finalizado ou não. Os resultados apresentam uma economia de 57% no tempo total de codificação com uma perda de qualidade de imagem de 1,21%.

O trabalho desenvolvido por (SALDANHA et al., 2022a), propõe uma solução configurável para a etapa de particionamento de blocos utilizando o classificador *Light Gradient Boosting Machine* (LGBM), com o objetivo de pular maneiras de dividir os blocos que provavelmente não seriam escolhidos como a melhor. Foram criados e treinado cinco modelos de classificação (um para cada maneira de dividir os bloco) baseados em informações de textura e de codificação da CU. Considerando os cinco pontos de controle, a proposta pode reduzir o tempo total de codificação de 35,22% até 61,34% com uma uma perda de eficiência de codificação variando de 0,46% até 2,43%.

O trabalho desenvolvido por (SALDANHA et al., 2022b) apresenta uma proposta utilizando um classificador Árvore de Decisão para acelerar a etapa de transformada da predição intra no VVC. Os classificadores irão prever quando os processos de avaliações *Multiple Transform Selection* (MTS) e *Low-Frequency Non-Separable Transform* (LFNST) provavelmente podem ser evitados. Os resultados experimentais mostram uma economia no tempo de codificação de 11% com uma perde de 0,43% na qualidade de imagem.

Após análise dos trabalhos, se constatou que existem apenas três trabalhos

buscando reduzir o esforço computacional do VVC com foco na etapa de AME, conforme mostrado na Tabela 1 coluna "**Etapa VVC**". Desses três trabalhos, dois são baseados em análise estatística (Est.) usando teorema de Bayes e apenas um é baseado em Aprendizado de Máquina (ML), com o método Classificador Florestas Aleatórias (RFC). Os demais trabalhos apresentam propostas para redução do esforço computacional baseadas em Aprendizado de Máquina utilizando métodos classificadores, em diferentes etapas do VVC, principalmente na etapa de particionamento de blocos (QTMT), obtendo bons resultados como mostrado nas colunas "**RTC**" (Percentual de Redução no Tempo de Codificação) e "**PQI**" (Percentual de Perda de Qualidade de Imagem). Dos oito trabalhos listado que utilizam ML, seis utilizam o método classificador de Florestas Aleatórias, um utiliza o método classificador *Light Gradient Boosting Machine* (LGBM) e um utiliza o método classificador Árvores de Decisão (DT). É possível ainda perceber analisando a coluna "**Versão VTM**" que há uma grande variação entre as versões utilizadas em cada trabalho, sendo a versão mais antiga 3.0 e a mais recente 10.0.

Para comparação do quão rápido são lançadas novas versões, nosso trabalho utiliza as versões 14 e 16.2 do VTM, e a ultima versão lançada até o momento da escrita desse trabalho é a VTM 18.1.

Tabela 1 – Comparativo entre os trabalhos relacionados.

Autor	**Técnica**	**Método**	**Etapa VVC**	**RTC**	**PQI**	**Versão VTM**
Park	Est.	T. Bayes	AME	5%	0,10%	3.0
Jung	Est.	T. Bayes	AME	5%	0,04%	10.0
Duarte	ML	RFC	AME	8%	0,18%	9.0
Amestoy	ML	RFC	QTMT	25%∼61%	0,4%∼2,2%	5.0
Zhang	ML	RFC	QTMT	54,91%	0,93%	4.0
CHEN	ML	RFC	QTMT	40,57%	1,14	7.0
LI	ML	RFC	QTMT	15,27%	0,07%	10.0
HE	ML	RFC	QTMT	57,00%	1,21	7.0
Saldanha a	ML	LGBM	QTMT	35%∼61%	0,4%∼2,4%	10.0
Saldanha b	ML	DT	Transf.	11%	0,43%	10.0

Fonte: Elaborado pelo autor.

4.1 Considerações Finais

Como já discutido anteriormente o VVC possui maior eficiência de compressão de dados que os padrões de codificação anteriores, porém esse ganho na compressão veio acompanhado de um aumento do esforço computacional causado pelas melhorias realizada nas ferramentas do codificador. A etapa de predição inter-quadros é uma das etapa que mais contribui para a compressão de dados, porém também representa um alto percentual do esforço computacional do codificador (SIQUEIRA; CORREA; GRELLERT, 2020).

Uma das novidades do VVC é a ferramenta de ME *Affine*, que possibilita a codificação de movimentos complexos como zoom e rotação com maior precisão do que os modelos de codificação translacionais. Porém essa maior eficiência de codificação traz junto uma elevação no custo computacional da etapa da ME conforme descrito por (PARK; KANG, 2019). Outra percepção foi que ainda exitem poucos trabalhos com foco na redução do esforço computacional da ME *Affine*.

Considerando esses trabalhos relacionados ainda há espaço para desenvolver técnicas para redução da complexidade no VVC. Assim, este trabalho traz como objetivo geral reduzir a complexidade computacional do VVC com foco na etapa da ME *Affine*, com os seguintes objetivo específicos:

- Apresentar as etapas de ME Unidirecional, Bidirecional e *Affine* do padrão VVC;
- Analisar do impacto da complexidade da AME na predição inter-quadros para cada tamanho de CU;
- Desenvolver soluções que diminuam a complexidade computacional do VVC;

Então nos capítulos 6 e 7 são apresentadas duas proposta com objetivo de realizar a redução do esforço computacional da etapa ME *Affine*, uma das propostas é baseada em uma implementação em hardware, e a outra baseada em otimização de software utilizando Aprendizado de Máquina.

5 METODOLOGIA

Nessa sessão é descrita a metodologia utilizada para obter os resultados das propostas desenvolvidas nesse trabalho. A análise dos resultados foi realizada seguindo as recomendações da Condição Comum de Testes do VTM (em inglês VTM *Common Test Conditions* CTC) (BOSSEN et al., 2020), documento que orienta a maneira de proceder durante os experimentos permitindo que possam ser comparados com outros trabalhos. Todas as sequências de vídeo utilizadas nas propostas foram codificadas quatro vezes, uma para cada Parâmetro de Quantização (em inglês *Quantization Parameter* - QP), sendo os valores 22, 27, 32 e 37 e com a configuração de *Random Access* habilitada. Para os testes foi utilizado o software Modelo de Teste do VVC (em inglês VVC *Test Model* - VTM) (VTM, 2022), esse software possui todas as ferramentas implementadas no codificador e permite que modificações sejam realizadas e testadas. A CTC ainda disponibiliza sequências de vídeo padrão para que os testes possam ser realizados. A Tabela 2 mostra estas sequências de vídeo e suas características.

Tabela 2 – Sequências de vídeos reacomodadas pela CTC

Classe	**Sequências**	**Resolução**	**Nº Quadros**	**Taxa Quadros**
A1	Tango2	3840×2160	294	60
A1	FoodMarket4	3840×2160	300	60
A1	Campfire	3840×2160	300	30
A2	CatRobot	3840×2160	300	60
A2	DaylightRoad2	3840×2160	300	60
A2	ParkRunning3	3840×2160	300	50
B	MarketPlace	1920×1080	600	60
B	RitualDance	1920×1080	600	60
B	Cactus	1920×1080	500	50
B	BasketballDrive	1920×1080	500	50
B	BQTerrace	1920×1080	600	60
C	RaceHorses	832×480	300	30
C	BQMall	832×480	600	60
C	PartyScene	832×480	500	50
C	BasketballDrill	832×480	500	50
D	RaceHorses	416×240	300	30
D	BQSquare	416×240	600	60
D	BlowingBubbles	416×240	500	50
D	BasketballPass	416×240	500	50
F	ArenaOfValor	1920×1080	600	60
F	BasketballDrillText	832×480	500	50
F	SlideEditing	1280×720	300	30
F	SlideShow	1280×720	500	20

Fonte: Elaborado pelo autor.

Para a análise de desempenho do codificador duas métricas são analisadas: **(i)**

tempo de codificação e **(ii)** eficiência da codificação como descrito pela CTC. Os resultados para essas métricas são obtidos através da comparação dos dados entre o VTM original (sem nenhuma modificação) e o VTM alterado (modificação realizada de acordo com proposta).

Para o cálculo da diferença de tempo de codificação (DTC) entre o VTM original e o VTM alterado é utilizada a Equação 5.1, onde T_O é o tempo de codificação do VTM original e T_A é o tempo de codificação do VTM alterado. Se o resultado dessa equação retornar um valor positivo, indica que o VTM alterado, realizou a codificação em um tempo menor que o VTM original (diminuiu o esforço computacional), caso ocorra o contrário, o valor seja negativo, indica que o VTM alterado levou mais tempo para executar que o VTM original (aumentou o esforço computacional).

$$DTC = \frac{T_O - T_A}{T_O} \times 100 \tag{5.1}$$

Esse mesmo cálculo pode ser utilizado para obtenção da diferença de tempo de ferramentas específicas do codificador, para isso devem ser utilizado os tempos exclusivos da ferramenta que se deseja obter essa informação.

Para a análise da eficiência de codificação foi utilizada a métrica *Bjontegaard Delta-rate* (BD-BR) (BJONTEGAARD, 2001). Essa métrica apresenta a taxa de compressão considerando que ambos vídeos possuem a mesma qualidade visual, fazendo uma relação entre a taxa de *bits* gerada e a qualidade do vídeo.

Buscando uma melhor compreensão da métrica BD-BR, define BS_O como o *bistream* gerado por uma sequência de vídeo codificada pelo VTM original e BS_A o *bitstream* gerado pela mesma sequência de vídeo codificada pelo VTM alterado. O aumento do percentual do BD-BR indica um acréscimo na taxa de *bits* do BS_A em relação ao BS_O, mantendo a mesma qualidade objetiva do vídeo. Conclui-se que o acréscimo no BD-BR, significa que a alteração implementada no VTM gera resultados negativos na eficiência de codificação.

6 HEURÍSTICA CONFIGURÁVEL *HARDWARE-FRIENDLY* PARA PREDIÇÃO INTER-QUADROS DO VVC

Como já mencionado o ganho em eficiência de compressão de dados veio acompanhado de um aumento no custo computacional do VVC. Uma das restrições para utilização em larga escala do VVC atualmente é o consumo de energia, principalmente em dispositivos móveis. Então, este capítulo descreve o desenvolvimento de uma heurística configurável, visando a economia de energia na etapa de predição inter-quadros do VVC e de fácil implementação em hardware, característica que chamamos de *hardware-friendly*.

O trabalho apresentado nessa sessão foi resultado da escrita do artigo publicado no 29º IEEE *International Conference on Electronics Circuits and Systems* (ICECS) (LOOSE et al., 2022).

6.1 Método Proposto

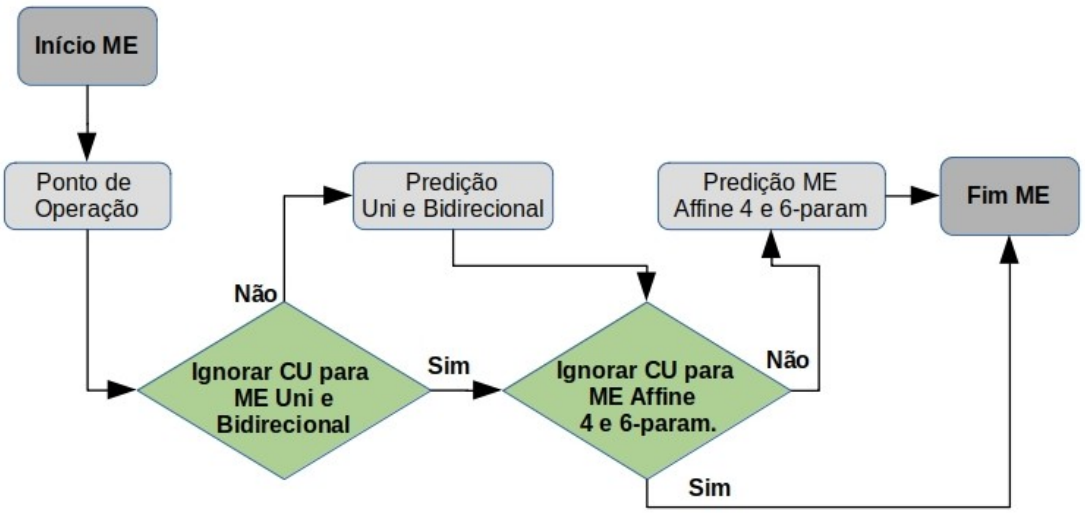
O trabalho propõe uma heurística *hardware-friendly* configurável para as etapas da ME Unidirecional, Bidirecional e *Affine* no VVC. A heurística é baseada na restrição para determinados tamanhos de CUs, de acordo com a resolução do vídeo e o ponto de operação. São definidos três pontos de operação de acordo com o nível da redução do consumo de energia: Mínimo (Min), Intermediário (Int) e Máximo (Max). Naturalmente quando maior for a economia de energia maior é a perda de qualidade de imagem.

O fluxograma da heurística é apresentado na Figura 15. Inicialmente o ponto de operação deve ser configurado. Essa configuração poderá ser fixada durante o desenvolvimento do projeto ou em tempo de execução. No primeiro caso é aplicado quando o ponto de operação é definido antes da implementação no hardware. No segundo caso é indicado quando o ponto de operação necessita ser definido no momento da execução, por exemplo, considerando o nível de energia que se deseja economizar de um dispositivo móvel.

Os tamanhos de CUs permitidos para cada etapa da ME é definido pelo ponto de operação e a resolução do vídeo. A primeira verificação define se determinado tamanho de CU deve per ignorado pelas etapas ME Unidirecional e Bidirecional. Se o tamanho de CU for ignorado, as etapas ME Unidirecional e Bidirecional não são aplicadas para esse tamanho de CU, e ambas as unidades podem ser desligadas. Por outro lado se o tamanho de CU não for pulado, ela é executada integralmente pelas etapas ME Unidirecional e Bidirecional. Após, uma segunda verificação define se determinado tamanho de CU está disponível para etapa da ME *Affine*. Se a etapa ME *Affine* for ignorada, então a unidade poderá ser desligada, caso contrário a ME *Affine* é executada integralmente.

Como se pode concluir observando a Figura 15, a heurística proposta é fácil de implementação em hardware. Quando os tamanhos de CUs pulados por cada unidade forem definidos durante a implementação do projeto, tanto a área de hardware como energia podem ser economizados. Em tempo de execução duas comparações simples devem

Figura 15 – Fluxograma da heurística proposta.



Fonte: Próprio Autor.

ser implementadas (nível de redução de energia e a resolução do vídeo) para definir qual unidade vai ser desligada para economizar energia de acordo com o ponto de operação e resolução do vídeo.

Esse trabalho foi desenvolvido em um alto nível de abstração, sem a utilização de um hardware específico, e onde o consumo de energia foi estimado se baseando no tempo de codificação de cada tamanho de CU, ou seja, quanto mais tempo processando maior consumo de energia. E o ganho na redução do consumo de energia irá ocorrer no momento que a unidade de processamento para determinado tamanho de CU for desabilitada, ignorando o processamento da CU, reduzindo o consumo de energia.

6.2 Extração das *Features*

As *features* consideradas para esse trabalho são o tempo de codificação e a usabilidade de cada tamanho de CU. O tempo de codificação representa o intervalo de tempo que determinado tamanho de CU leva para executar em cada etapa da ME. A usabilidade nesse trabalho é representada pela quantidade de vezes que cada tamanho de CU é selecionada com o melhor custo RD (BROSS et al., 2021b) em cada etapa da ME, ou seja, de todas as vezes que ela foi testada para aquela etapa, quantas vezes ela foi selecionada como a CU a ser predita. O tempo de codificação é utilizado como uma métrica aproximada para consumo de energia.

A ideia principal é utilizar as *features* para encontrar a melhor combinação de tamanhos de CU para alcançar a redução de tempo de codificação esperada com o menor impacto possível na qualidade de imagem. A premissa é que quanto menos vezes o tamanho da CU possuir o menor custo RD (menos usabilidade), menor é o impacto de ignorar este

tamanho de CU no processo de codificação. As *features* são extraídas considerando as etapas da ME em dois grupos: **(i)** ME Unidirecional e Bidirecional e **(ii)** ME *Affine* com 4 e 6 parâmetros. Os grupos foram criados considerando a similaridade das operações para essas etapas do codificador.

Foram selecionadas 15 sequências de vídeo de maneira aleatória, separadas em três resoluções: HD (1280×720), FHD (1920×1080), e 4K UHD (3840×2160). As sequências selecionada foram: HD foram *Dark*, *Netflix DrivingPOV*, *Vidyo4*, *Netflix DinnerScene*, e *KristenAndSara*. FHD foram *Netflix TunnelFlags*, *Jockey*, *Beauty*, *Touchdown Pass*, e *Rush Field Cuts*. Por fim, 4K UHD foram *ToddlerFountain*, *SunBath*, *Lips*, *BuildingHall2* e *Netflix Dancers*. Cada sequência de vídeo foi executadas quatro vezes, uma para cada QP (22, 27, 32 e 37). Utilizando os primeiros 32 *frames* de cada vídeo.

As *features* foram extraídas para os tamanhos de CUs (largura×altura) permitidos em cada etapa da ME: 27 tamanhos de CUs para ME Unidirecional e Bidirecional e 12 para ME *Affine*. A Tabela 3 apresenta todos os possíveis tamanhos de CUs para cada etapa da ME.

Tabela 3 – Tamanhos de CUs Permitidas para ME Unidirecional, Bidirecional e *Affine*

ME Uni – Bidirecional			**ME Affine**	
4×8	16×4	32×64	16×16	64×16
4×16	16×8	64×4	16×32	64×32
4×32	16×16	64×8	16×64	64×64
4×64	16×32	64×16	32×16	64×128
8×4	16×64	64×32	32×32	128×64
8×8	32×4	64×64	32×64	128×128
8×16	32×8	64×128		
8×32	32×16	128×64		
8×64	32×32	128×128		

Fonte: Elaborado pelo autor.

6.3 Definição das Unidade de Codificação (CUs) Ignoradas

Na etapa de análise dos dados um algoritmo de força-bruta foi criado para testar todas as combinações entre os 27 tamanhos de CU permitidos para ME Unidirecional e Bidirecional e entre os 12 tamanhos de CU permitidos pela ME *Affine*. Para cada combinação de tamanho de CU e cada resolução, o algoritmo soma seus respectivos percentuais de tempo de codificação e usabilidade. Para cada meta, a combinação de CU com a menor usabilidade que atingiu ou ultrapassou a meta é selecionada para ser ignorada, considerando os dois grupos de etapas da ME (como mostrado nas Tabelas 4 e 5). A suposição básica é que quanto menor é a usabilidade de um tamanho CU (ou grupo de CUs) ignorada, menor tende a ser a perda de qualidade de imagem. Por exemplo, na

Tabela 5, para a resolução (Res) HD e ponto de operação (OP) Int, os tamanhos de CUs ignorados são 16×64 e 128×64, pois esse conjunto de tamanhos de CUs representa o menor percentual de usabilidade para alcançar uma redução de 15% no tempo da total da predição inter-quadros.

Tabela 4 – Tamanhos de CUs Ignoradas: ME Unidirecional e Bidirecional

Res	**OP**	**Blocos**	**% Te**	**% Us**
HD	**Min**	4×16, 4×64, 8×8	5,2	2,58
	Int	4×16, 4×64, 16×4, 128×128	15,04	9,3
	Max	4×8, 8×4, 4×16, 8×8, 4×32, 16×4, 4×64, 32×4, 128×128	20,2	12,55
FHD	**Min**	4×16, 4×64	5,06	2,58
	Int	4×16, 4×64, 8×8, 16×4, 128×64	15,4	8,92
	Max	4×8, 8×4, 4×16, 8×8, 4×32, 16×4, 32×4, 128×128	20,03	12,33
4K	**Min**	4×16, 8×8	5,82	2,78
	Int	4×16, 4×32, 4×64, 8×8, 128×64	15,32	8,77
	Max	4×8, 8×4, 4×16, 8×8, 4×64, 16×4, 128×128	20,14	11,75

Fonte: Elaborado pelo autor.

Tabela 5 – Tamanhos de CUs Ignoradas: ME Affine

Res	**OP**	**Blocos**	**% Te**	**% Us**
HD	Min	16×64	5,06	4,66
	Int	16×64, 128×64	15	9,43
	Max	16×64, 64×16, 128×64	20,29	16,14
FHD	Min	16×64	5,65	5,53
	Int	16×32, 64×128	15,13	14,6
	Max	64×128, 128×128	20,35	11,01
4K	Min	64×16	5,3	5,16
	Int	128×128	15,82	8,21
	Max	16×16, 128×128	20,6	14,53

Fonte: Elaborado pelo autor.

A heurística foi avaliada utilizando três pontos de operação, chamados de Mínimo (Min), Intermediário (Int) e Máximo (Max), em referência ao percentual redução do consumo de energia que se deseja alcançar em cada meta de redução do tempo de codificação. Nesse caso as metas são definidas como 5% (Min), 15% (Int) e 20% (Max) para os dois grupos de etapas da ME, porem outras metas podem ser definidas permitindo um maior nível de configurabilidade ou maior nível de economia de energia. Os valores que representam o percentual de redução de energia (pontos de operação) foram definidos de maneira empírica, representando níveis de redução do consumo de energia.

As tabelas 4 e 5 apresentam os tamanhos de CUs ignoradas, o percentual do tempo de codificação possível de redução ($\%Te$) e o percentual de usabilidade ($\%Us$) para cada ponto de operação (OP) e cada resolução (Res) suportada para **(i)** ME Unidirecional e Bidirecional e **(ii)** ME *Affine*, respectivamente. Como discutido anteriormente a heurística foi especializada para três resoluções de vídeo: HD, FHD e 4K UHD. Como se pode concluir observando a Tabelas 4 e 5, o número de tamanhos de CUs ignoradas varia de acordo com o ponto de operação, a resolução do vídeo e etapa da ME. Isso ocorre exatamente porque a heurística foi definida através de uma exaustiva avaliação da melhor combinação de tamanhos de CU para cada caso.

6.4 Resultados

A heurística foi desenvolvida com os respectivos pontos de operação Min, Int e Max, para cada um dos dois grupos da ME: **(i)** Unidirecional e Bidirecional e **(ii)** *Affine*. A implementação permite a avaliação experimental da redução do tempo de codificação e perda de eficiência na qualidade de imagem considerando os pontos de controle e a resolução dos vídeos, para cada grupo da ME. A metodologia utilizada para análise dos resultados foi descrita no Capítulo 5.

A proposta foi implementada no software de referência VTM 14.0 (A. CHEN J., 2021), executado em um servidor possuindo o sistema operacional Ubuntu 16.04.5, processador Intel Xeon E5-2650v4 2.20GHz com 48 GB de RAM. Para este trabalho foram selecionadas oito sequências de vídeo separadas por resolução, duas HD, três FHD e três 4K-UHD, conforme mostrado na Tabela 6. Cada sequência de vídeo foi executada para os 32 quadros inciais.

Tabela 6 – Sequências de vídeos utilizadas para análise dos resultados

Sequência	**Classe**	**Resolução**
SlideShow	F	1280×720 – HD
SlideEditing	F	1280×720 – HD
RitualDance	B	1920×1080 – FHD
Cactus	B	1920×1080 – FHD
BQTerrace	B	1920×1080 – FHD
Tango2	A	3840×2160 – 4K
FoodMarket4	A	3840×2160 – 4K
CatRobot	A	3840×2160 – 4K

Fonte: Elaborado pelo autor.

Esse trabalho apresentada uma estimação do ganho de economia de energia por ponto de operação e resoluções dos vídeos da heurística proposta. A estimação foi realizada com um alto nível de abstração e sem o desenvolvimento em um hardware específico, mas considerando a adaptabilidade do tempo de execução. Dois pontos são considerados:

(i) quando cada etapa da ME tem uma unidade de processamento para processar cada tamanho de CU (totalmente paralelo) e **(ii)** quando cada etapa da ME possui uma unidade de processamento genérica utilizada para processar todos os tamanhos de CU (totalmente sequencial). No caso do processamento paralelo, para cada tamanho de CU ignorada é necessário desligar a unidade operacional, assim reduzindo o consumo de energia. No caso sequencial, para cada tamanho de CU ignorada implica em menos ciclos de uso da unidade de processamento, assim aumentando a economia de energia. Então, o ganho no consumo de energia é proporcional ao número de tamanhos de CU ignoradas, independente do nível de paralelismo. A estimativa de redução de energia considerou que todas as quatro etapas da ME (Unidirecional, Bidirecional e *Affine* com quatro ou seis parâmetros) consomem a mesma energia, por simplicidade. Então a estimação do ganho de energia considera o percentual de tamanhos de CUs ignoradas em cada caso, em comparação com o total de tamanhos de CUs originalmente disponíveis no VVC: 27 para ME Unidirecional e Bidirecional e 12 para ME *Affine*.

A Tabela 7 apresenta os resultados alcançados incluindo a média de redução de tempo para Unidirecional e Bidirecional (TR_{UB}), a media de redução de tempo da ME *Affine* (TR_{AF}), o total de estimado de redução no consumo de energia baseado no tempo de codificação ($EERC$), e o total de perda de eficiência de compressão, usando o método *Bjøntegaard-Delta bitrate* ($BDBR$) (BJONTEGAARD, 2001). Esses resultados são apresentados na Tabela 7 por resolução (Res) e ponto de operação (OP), que são representados por ganhos ou perdas quando comparada a heurística proposta e o VTM original.

Tabela 7 – Resultados da redução de tempo, estimação da redução do consumo de energia e o aumento do BDBR

OP	**Res**	**TR**$_{UB}$ (%)	**TR**$_{AF}$ (%)	**EECR** (%)	**BDBR** (%)
	HD	3,57	6,73	10,26	0,04
Min	FHD	10,56	4,70	7,69	0,19
	4K UHD	6,19	8,12	7,69	0,11
	HD	12,55	14,15	15,39	0,35
Int	FHD	14,50	10,23	17,95	0,25
	4K UHD	11,12	16,81	15,39	0,30
	HD	16,09	19,55	30,77	1,00
Max	FHD	26,50	14,75	25,64	0,90
	4K UHD	15,68	22,71	23,08	0,44

Fonte: Elaborado pelo autor.

Os resultados da Tabela 7 mostram que, em todos os casos, a heurística proposta alcança importante economia de tempo e energia com baixo impacto em termos de perda de qualidade de imagem. Ainda, os resultados da Tabela 7 mostram que a heurística

proposta permite dimensionar a economia de energia de acordo com os pontos de operação. O melhor percentual de economia de energia é obtido com o ponto de operação Max em todos os casos, como esperado, alcançando mais de 30% de redução no consumo de energia para sequência de vídeos HD, para um aumento de 1% no BD-BR. Por outro lado, como esperado uma baixa perda de eficiência de codificação é alcançada no ponto de operação Min e novamente as sequências HD devem ser destacadas, pois para este ponto de operação as sequências HD atingiram a maior redução de consumo de energia (10,26%), com o menor impacto BD-BR (apenas 0,04%) entre todos os pontos de operação. Outra observação é que as reduções de tempo de codificação apresentam algumas imprecisões em comparação com os metas originais para os dois grupos de etapas ME. Ainda assim, mesmo com tais imprecisões, as reduções de tempo de codificação variam de forma consistente com os pontos de operação, que foi o objetivo principal do trabalho.

Não é possível fazer comparações com trabalhos relacionados de maneira equivalente, já que nenhum dos trabalhos na literatura focam em soluções de fácil implementação em hardware ou solução configuráveis. Como falado anteriormente, existem alguns trabalhos focando na redução do tempo de codificação da predição inter-quadros do VVC usando uma variedade de técnicas, como os trabalhos (DUARTE et al., 2022), (JUNG; JUN, 2021) e (PARK; KANG, 2019), que miram na ferramente ME *Affine* do VVC. Estes trabalhos relacionados não estão preocupados com os problemas da implementação de hardware e então seus métodos não são fáceis de serem implementados em hardware. Em uma aplicação real onde a solução em hardware é obrigatória, como na maioria das aplicações que suportam codificadores de vídeo atuais, as soluções apresentadas nestes trabalhos não são diretamente aplicáveis. Esses trabalhos relacionados também não suportavam nenhum nível de configurabilidade, o que também é um recurso importante para aplicativos do mundo real, principalmente aqueles executados em dispositivos alimentados por bateria. Além disso, esses trabalhos utilizaram diferentes versões do VTM, evitando também comparações diretas.

Com um foco diferente dos trabalhos relacionados, a heurística proposta aqui alcançou resultados interessantes. Enquanto trabalhos relacionados tem foco total em otimizações de software, esta heurística amigável a hardware e configurável alcançou quase 23% da redução do tempo de codificação para a ferramente ME *Affine* com um aumento de 0,44% de BDBR quando considerando as sequencias 4K UHD e o ponto de operação Max. Considerando apenas as otimizações de software, estes três trabalhos relacionados (DUARTE et al., 2022), (JUNG; JUN, 2021) e (PARK; KANG, 2019) alcançaram reduções de tempo maiores (de 33% (JUNG; JUN, 2021) até 47% (DUARTE et al., 2022)) e menores resultados de BD-BR (de 0,04% (JUNG; JUN, 2021) até 0,18% (DUARTE et al., 2022)) para ME *Affine*. Mas estes trabalhos relacionados focam somente em uma ferramenta ME e não consideram problemas de hardware e nem providenciam suporte configurável. Além disso, como esperado, estes trabalhos não discutem implementação em hardware ou

problemas de consumo de energia.

6.5 Conclusão

Foi apresentada uma heurística configurável e amigável a hardware visando redução do consumo de energia da pedição inter-quadros do VVC. A heurística foi definida usando três pontos de operação e foi especializada para três resoluções de vídeos nas etapas da ME Unidirecional, Bidirecional, e *Affine*. Os resultados dos experimentos mostraram reduções do consumo estimado de energia entre 7,69% e 30,77% com uma perda de eficiência de codificação variando entre 0,04% e 1%. Estes resultados são expressivos, principalmente ao considerar que a maioria dos aplicativos de vídeo estão sendo executados em dispositivos alimentados por bateria hoje em dia. Até onde se sabe esse é o primeiro trabalho na literatura que apresenta uma solução *hardware-friendly* configurável abordando a redução do consumo de energia da predição inter-quadros VVC.

7 RÁPIDA ESTIMAÇÃO DE MOVIMENTO *AFFINE* VVC BASEADA EM APRENDIZADO DE MÁQUINA

VVC é o parão estado-da-arte, desenvolvido para solucionar as demandas atuais e que não são atendidas pelo padrão antecessor HEVC. A redução do volume de dados gerado pelos conteúdos de vídeo é uma das principais demandas, estima-se que a taxa de redução de *bits* em relação ao HEVC já esteja próxima a 50%, porem isso é penalizado com o aumento do esforço computacional necessário para essa tarefa. O aumento do custo para realizar a codificação é decorrente das melhorias e inclusões de ferramentas mais eficientes ao codificador VVC. Uma destas ferramentas é a ME *Affine*. Como já apresentado anteriormente, ela é responsável por codificar movimentos não-translacionais com maior precisão que as técnicas utilizadas para movimentos translacionais, porém como consequência um esforço computacional maior é necessário, podendo a AME representar até 54% de toda a etapa de Estimação de Movimento (ME). Esse trabalho apresenta uma proposta para reduzir o esforço computacional do codificador VVC, com foco na etapa de ME *Affine*.

Como visto nos trabalhos relacionados, vários deles aplicam técnicas de Aprendizado de Máquina utilizando modelos classificadores em suas soluções, a maioria aplica as técnicas principalmente na etapa de particionamento de blocos e apenas um na etapa da AME. Sendo assim, a proposta é baseada na otimização de software utilizando técnicas de Aprendizado de Máquina com método classificador Florestas Aleatórias aplicada na etapa de Estimação de Movimento *Affine* (AME).

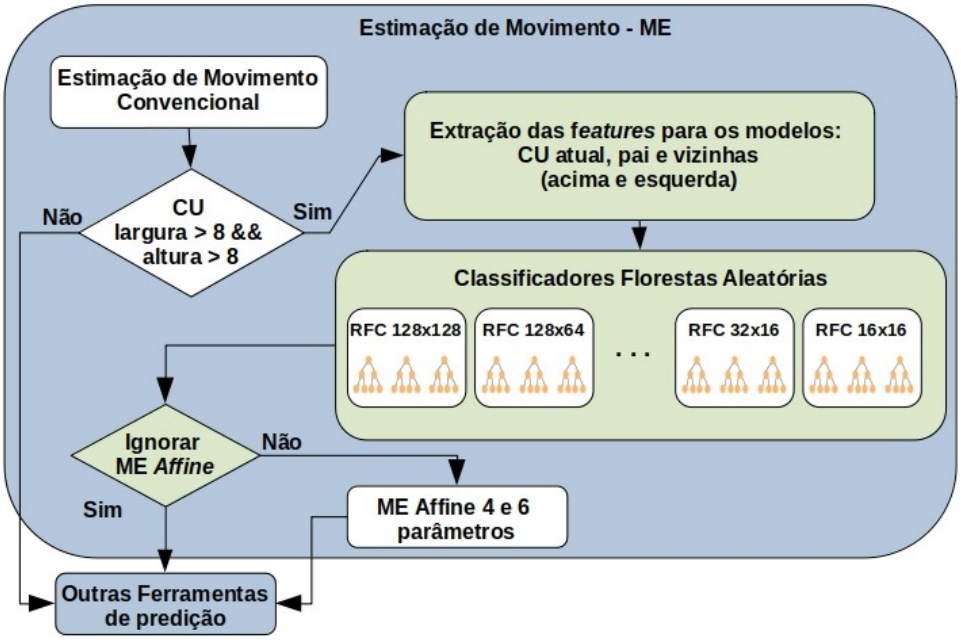
7.1 Método Proposto

O trabalho desenvolve um modelo de Aprendizado de Máquina com a técnica de classificação Florestas Aleatória (em inglês *Random Forest Classification* - RFC) para otimizar o tempo de codificação do VVC, focando da etapa de Estimação de Movimento *Affine* (AME). A proposta explora as características de cada tamanho de CU e cria modelos para decidir se determinado tamanho de CU deverá ser ignorado completamente pela AME ou não. A Figura 16 apresenta um fluxograma, onde as partes destacadas em verde claro representam a solução proposta.

Inicialmente foi verificado se o tamanho da CU possui largura e altura maiores que 8, essa restrição já é definida originalmente pelo codificador, se essa condição for satisfeita, as *features* são extraídas da CU atual e também das CUs pai e vizinhas à CU atual. Na próxima etapa, o modelo de RFC é escolhido de acordo com o tamanho da CU atual, existirá um modelo de RFC para cada tamanho de CU suportado pela AME. O modelo RFC selecionado recebe um vetor contendo todas as *features* coletadas. As *features* são classificadas pelo RFC retornando como saída a resposta se o CU deverá ser ignorada pela AME ou não. Caso a AME seja ignorada, ela não é executada para aquela CU, caso

contrário a AME é executada normalmente para as etapas *Affine* com 4 e 6-parâmetros.

Figura 16 – Fluxograma do método proposto.



Fonte: Próprio Autor.

O diferencial desse trabalho em relação aos trabalhos encontrados na literatura está na criação de um modelo de RFC para cada tamanho de CU suportado pela AME, assim podendo ser exploradas as características de processamento de cada tamanhos de CU de forma independentemente.

7.2 Mineração de dados

O processo de mineração de dados consiste de extrair informações do codificador durante a sua execução. Como o método utilizado é de aprendizado supervisionado, é necessário que se possua previamente um conjunto de instâncias para criação e treinamento do modelo RFC. Como já mencionado anteriormente, uma instância é representada por um conjunto de variáveis de entrada chamadas de *features* e uma variável de saída que representa o resultado da classificação, chamada de classe ou alvo.

Para a extração das *features* foram selecionadas 10 sequências de vídeo de maneira aleatória e de diferentes resoluções, obtidas nos *datasets* UVG (MERCAT; VIITANEN; VANNE, 2020) e NETVC (DAEDE; NORKIN; BRAILOVKKIY, 2018) e JVET (BOSSEN et al., 2020). Duas sequências 4K-UHD, duas sequências FullHD, duas sequência HD e por último quatro sequências CIF. A Tabela 8 mostra a descrição das sequência de vídeo por resolução. Cada sequência de vídeo foi codificada quatro vezes, uma para cada QP (22,

27, 32 e 37) conforme definido pela CTC VVC, utilizando a configuração *Random Access* habilitada.

Tabela 8 – Sequências de vídeo para extração das *features*.

Sequência	**Resolução**	**Dataset**
Lips	3840×2160 – 4K	UVG
Beauty	3840×2160 – 4K	UVG
Rush field cuts	1920×1080 – FullHD	NETVC
Netflix TunnelFlag	1920×1080 – FullHD	NETVC
Vidyo4	1280×720 – HD	NETVC
KristenAndSara	1280×720 – HD	JVET
Highway	352 x 288 – CIF	NETVC
Foreman	352 x 288 – CIF	NETVC
Container	352 x 288 – CIF	NETVC
Coastguard	352 x 288 – CIF	NETVC

Fonte: Elaborado pelo autor.

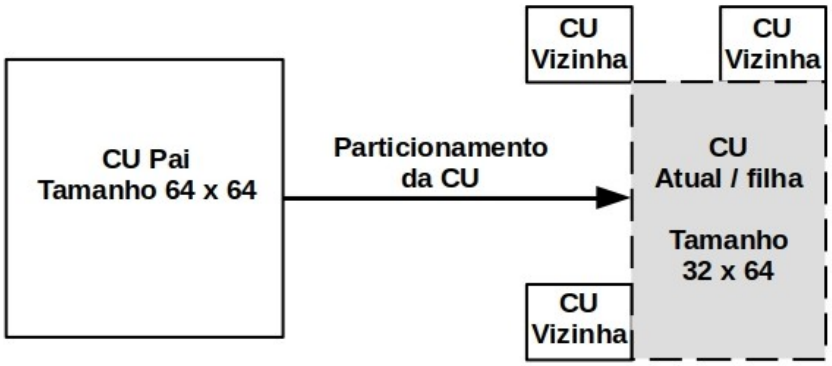
Os critérios utilizados para selecionar as *features* foram:

- Obter informações que são geradas durante a execução normal do codificador evitando processamentos adicionais, ou seja, evitar processamento de informações para gerar novos dados;
- Da CU atual, extrair *features* geradas durante a execução da Estimação de Movimento Convencional (CME) e que são utilizadas pelo codificador para realizar o cálculo do custo RD (custo utilizado pelo codificador para definir qual etapa da ME codificou a CU com melhor eficiência);
- Das CUs previamente codificadas (pai e vizinhas), extrair informações de qual etapa da ME a CU foi codificada com a melhor eficiência (se foi AME ou não), pois segundo o trabalho de (JUNG; JUN, 2021), quando uma CU pai é codificada com AME, é grande a probabilidade da CU filha também ser.

Baseada na abordagem descrita pelo trabalho de (JUNG; JUN, 2021) que descreve a relação entre as CU pai e filha, para este trabalho incluímos essa abordagem para a CUs vizinhas (acima e a esquerda). Nesse sentido *features* das CUs vizinhas também foram extraídas devido a proximidade com a CU atual, sendo que as características de movimento aplicadas a CU vizinha podem ser similares a da CU atual. A Figura 17 apresenta o posicionamento das CUs em relação a CU atual. Onde a CU pai representa a CU que deu origem a CU atual após o particionamento, e as CUs vizinhas ficam localizada acima e a esquerda da CU atual.

Conforme os critérios definidos acima, foram selecionas 39 *features*, descritas na Tabela 9. Em algumas *features* há mais de um valor associado. A quantidade de

Figura 17 – Posicionamento das CUs em relação a CU Atual.



Fonte: Próprio Autor.

valores associados é descrita pela coluna *Qtd*. Por exemplo, a *feature cu_pos* possui 2 valores representados por x e y, indicando a posição da CU dentro do quadro. *Mv_uni* é representado por 4 valores, para cada posição do vetor de movimento (x e y), para cada uma das listas (Lista 0 e Lista 1).

Os valores de cada *feature* são salvos em um arquivos após a codificação de cada tamanho de CU. São armazenadas apenas a informações para os 12 tamanhos de CU permitidas pela AME, são elas: 16×16, 16×32, 16×64, 32×16, 32×32, 32×64, 64×16, 64×32, 64×64, 64×128, 128×64, 128×128.

Finalizada a extração das *features* após a codificação de todas as sequências de vídeos, as instâncias foram separadas por tamanhos de CU, resultando em 12 *datasets*, um para cada tamanho de CU permitido para AME. Para cada *dataset* todas as instância repetidas foram removidas. Para o balanceamento dos dados foi utilizada a biblioteca em *Python Imbalanced-learn* (LEMAÎTRE; NOGUEIRA; ARIDAS, 2017) através do método *under-sampling*, que consiste em eliminar aleatoriamente instâncias da classe majoritária, deixando todas as classes com o mesmo número de instâncias.

Nos modelos propostos duas classes são definidas a Classe 0 e a Classe 1. A Classe 0 representa as instâncias onde o melhor modo de predição escolhido não foi a AME. A Classe 1 representa as instância onde o melhor modo de predição escolhido foi a AME.

A Tabela 10 apresenta o número de instâncias extraídos para cada classe em cada tamanho de CU, antes do balanceamento e após o balanceamento. Como apresentado nas colunas “Classe 0 Desbal.” e “Classe 1 Desbal.” a diferença entre o número de instância em cada classe é bastante grande, por essa razão existe a necessidade do ajuste entre as classes, evitando que ocorra o *overfitting* dos modelos, ou seja, evitar que o modelo aprenda classificar melhor uma classe (por possuir mais instâncias) do que a outra.

Tabela 9 – Relação de *features* selecionadas.

Feature	**Qtd**	**Descrição**
QP	1	Parâmetro de quantização de entrada
cu_pos	2	Posição X e Y da CU dentro do quadro
bcw_index	1	Peso atribuídos as CUs para durante a etapa de bi predição
imv	1	Indica a precisão do vetor de movimento
mv_uni	4	Posição X e Y para o vetor de movimento gerado pela predição unidirecional para Lista 0 e Lista 1
mv_pred_uni	4	Posição X e Y para o vetor de movimento gerado pela predição AMVP unidirecional para Lista 0 e Lista 1
cost_mv_uni	2	Custo do vetor de movimento para a Lista 0 e Lista 1 da predição unidirecional
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
mv_pred_bi	4	Posição X e Y para o vetor de movimento gerado pela predição AMVP bidirecional para Lista 0 e Lista 1
cost_bi	1	Custo gerado pela predição bidirecional
bits_mv_bi	1	Bits gerados pela predição bidirecional
inter_dir_normal	1	Indica a direção da Predição Unidirecional (L0 ou L1) ou Bidirecional
atual_QP	1	QP que está sendo utilizado no momento
affine_pai	1	Indica se o bloco pai é codificado com AME
custo_pai	1	Custo gerado pela predição do bloco pai
imv_pai	1	Valor do IMV do bloco pai
aff_viz_esq_1	1	Indica se o vizinho a esquerda é codificado com AME
custo_viz_esq	1	Custo gerado pela predição do vizinho a esquerda
IMV_viz_esq	1	Valor do IMV do bloco vizinho à esquerda
aff_viz_acima	1	Indica se o vizinho acima é codificado como AME
custo_viz_aci	1	Custo gerado pela predição do vizinho acima
IMV_viz_aci	1	Valor do IMV do bloco vizinho acima
cu_atual_affine	1	Se a CU atual foi possui o melhor modo de predição como AME

Fonte: Elaborado pelo autor.

7.3 Desenvolvimento dos modelos RFC

Inicialmente foi realizada uma análise de cada tamanho de CU dentro da ME. Foram extraídos o tempo de codificação que cada tamanho de CU que foi processada pela ME, esses tempos foram extraídos separadamente para a Estimação de Movimento convencional (CME) e para Estimação de Movimento *Affine* (AME). Os resultados são mostrados na Figura 18. Outra informação coletada da ME foi o somatório de todas as CUs processadas pela AME e separadas por tamanho, sendo apresentada em percentual.

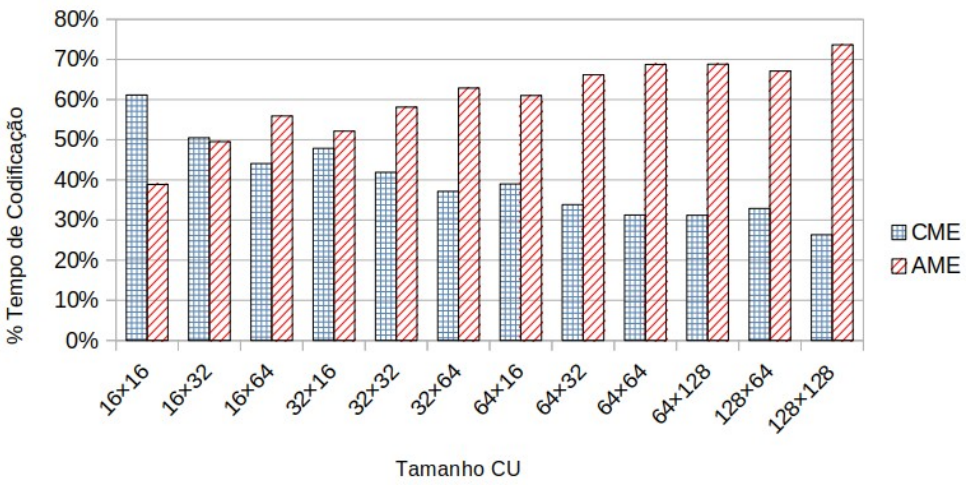
Tabela 10 – Quantidade de instâncias extraídas para cada tamanho de CU.

Tam. CUs	**Classe 0 Desbal.**	**Classe 1 Desbal.**	**Classe 0 Balanc.**	**Classe 1 Balanc.**
128x128	681318	90846	90846	90846
128x64	1072942	70106	70106	70106
64x128	1074495	71852	71852	71852
64x64	3803914	257292	257292	257292
64x32	7359213	447442	447442	447442
64x16	9552511	522657	522657	522657
32x64	7460291	459455	459455	459455
32x32	16005767	963191	963191	963191
32x16	14935958	1526067	1526067	1526067
16x64	9898160	548407	548407	548407
16x32	15007573	1552231	1552231	1552231
16x16	19679199	1511119	1511119	1511119

Fonte: Elaborado pelo autor.

Os resultados são mostrados na Figura 19.

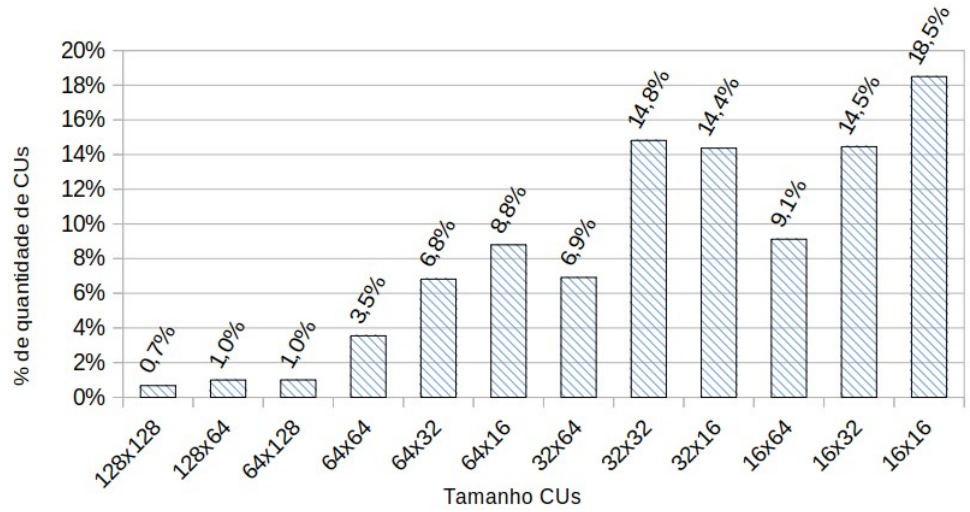
Figura 18 – Gráfico comparando tempos CME e AME.



Fonte: Próprio Autor.

Analisando a Figura 18, é possível perceber que em tamanhos de CUs que possuem maior número de amostras a AME apresenta um tempo maior de codificação em relação a CME. Por exemplo, a CU 128×128, mais de 70% do seu tempo de codificação é utilizado pela AME, já as CUs 16×16 e 16×32 o tempo de codificação da AME é inferior a CME. Por outro lado, analisando a Figura 19, CUs com números de amostra maiores apresentam uma quantidade menor de CU, já CUs com números de amostras menores apresentam quantidade maior de CUs. Por exemplo, CU 128×128 representa 0,7% de todas as CUs processadas, já a CU 16×16 representa 18,5% desse total.

Figura 19 – Gráfico da quantidade de CUs por tamanho.



Fonte: Próprio Autor.

A conclusão obtida com a analise das Figuras 18 e 19, foi que o tempo de codificação da AME é afetado diretamente pelo número de amostras da CU e não pela sua quantidade. Isso se deve à quantidade de sub-CUs (4×4) que cada CU vai gerar. Pois quanto maior o número de sub-CUs, maior o custo computacional para derivar os MVs, realizar o arredondamento e aplicar os filtros de interpolação. Após essa análise se optou por criar um modelo de RFC para cada tamanho de CU.

Após os dados estarem ajustados de acordo com o tamanho de CU e balanceados pelo número de classes, os modelos foram treinados utilizando a biblioteca em Python Scikit-learn (PEDREGOSA et al., 2011) de maneira *offline*, ou seja, eles são treinados antecipadamente utilizando um *dataset* estático. O modelo de Classificação Florestas Aleatórias permite a definição de hyper-parâmetros para ajuste do modelo, isso garante ao modelo uma maior precisão nos resultados. Para este trabalho foram utilizados três hyper-parâmetros, *criterion*, *max_depth* e *n_estimator*.

- *Criterion* define a qualidade das divisões dos nós das árvores de decisão que compõe a RFC (descrito anteriormente na sessão 3.1).

- *Max_depth* define a profundidade das árvores de decisão. Quanto mais rasa for a árvore maior é a possibilidade de gerar *underfitting* (não conseguir classificar corretamente as classes pois possui poucas informações) nos modelos, quando a árvore de decisão for muito profunda ela poderá gerar modelos com *overfitting* (o modelo fica muito especializado, apresenta bons resultados somente para instância de treinamento).

- *N_estimator* define o número de árvores de decisão que a floresta vai possuir,

quando maior o número de árvores maior é a precisão do modelo, porém mais lento se torna o processamento.

Após as primeiras etapas de treinamento, foi identificada três *features* com valor zero para *feature importance* retornadas pelo modelo de tamanho de CU 128×128, sendo elas: $affine\_pai$, $custo\_pai$ e imv_pai. Isso ocorre pois a CU de tamanho 128×128 é considerada uma CTU, sendo o primeiro bloco particionado, por isso não possuirá CU pai, então essa *features* foram removidas para esse modelo.

A definição dos valores atribuídos aos hyper-parâmetros foi realizada através de uma densa análise experimental, onde diferentes valores foram testados até encontrar os que retornassem os melhores resultados. Abaixo são listados os valores testados nos hyper-parâmetros $Criterion$, max_depth e $n\_estimator$:

- **Criterion**: testada apenas com o valor "Gini";
- **max_depth**: foram testados os valores variando entre 3 e 6, evitando que as árvores fossem muito rasas, com menos de 3 níveis, ou muito profundas com mais de 6 níveis. Árvores com profundidade maiores que 6 níveis seriam muito grandes (em número de linhas), dificultando a implementação no software de referência VTM;
- **max_depth:** foram testados com os valores 50, 100, 150, 200 e 250;

A Tabela 11 apresenta os valores escolhidos para os hyper-parâmetros e as configurações utilizadas para treinar os 12 modelos RFC utilizados nesse trabalho. A coluna IPC apresenta o número de instâncias, para cada uma das classes, usadas para treinar o modelo após o balanceamento das classes. Colunas MD e NE representam os valores definidos para os hyper-parâmetros max_depth e $n\_estimator$, respectivamente. O valor "Gini" foi atribuído para o hyper-parâmetros *criterion* em todos os modelos criados.

A Tabela 11 também apresenta os resultados das métricas utilizadas para avaliar o modelo, Acurácia, *Recall*, Precisão e *F1-Score* representados pelas colunas Ac , Re, Pre e $F1$ respectivamente. Para evitar *overfiting* dos modelos, conjuntos de dados diferentes foram utilizados para treinamento e testes. Durante o treinamento dos modelos foi priorizado que a métrica *Recall* possuísse valor maior ou igual a 90%, garantindo que ao menos 90% dos blocos com o melhor custo RD *affine* fossem testados, evitando a perda de qualidade de imagem. Para a métrica de Acurácia dos modelos, ela variou entre 76% e 82%, a métrica de Precisão ficou entre 24% e 57% e o *F1-Score* entre 0,37% e 0,69%. Com esses valores os modelos obtiveram resultados satisfatórios para a proposta apresentada.

Uma descrição do funcionamento dos modelos pode ser analisada observando a Figura 14, onde as **Instâncias de Entrada** são representadas pelas *features* contidas na Tabela 9, exceto a contida na última linha da tabela, cu_atual_affine que representa a variável alvo/classe. As *features* são encaminhadas para as diversas DTs que compõe a RFC, cada DT vai realizar de forma independente sua classificação de acordo com as

Tabela 11 – Configurações de treino e resultados.

Modelos	IPC	MD	NE	Ac	Re	Pre	F1
RF_128×128	90846	5	150	0,82	0,88	0,57	0,69
RF_128×64	70106	4	200	0,81	0,96	0,42	0,58
RF_64x128	71852	4	200	0,81	0,95	0,42	0,58
RF_64×64	257292	4	200	0,80	0,92	0,36	0,52
RF_64×32	447442	4	200	0,82	0,93	0,34	0,50
RF_64×16	522657	4	200	0,82	0,93	0,30	0,46
RF_32×64	459455	4	200	0,82	0,93	0,35	0,51
RF_32×32	963191	4	200	0,80	0,93	0,28	0,43
RF_32×16	1526067	4	100	0,79	0,90	0,30	0,44
RF_16×64	548407	4	200	0,82	0,94	0,31	0,46
RF_16×32	1552231	4	100	0,79	0,90	0,31	0,45
RF_16×16	1511119	4	200	0,76	0,92	0,24	0,37

Fonte: Elaborado pelo autor.

features recebidas e retornar o resultado para qual classe aquele conjunto de *features* foi classificada. O resultado de cada DT é recebido pelo algoritmo de votação, que irá analisar qual classe recebeu o maior número votos entre todas as DTs da RFC. E por fim o valor da variável alvo é retornado pelo algoritmo de votação indicando qual classe foi selecionada. Retornando Classe 0, indica que o tamanho de CU deve ser ignorado pela AME, retornando a Classe 1, indicada que o tamanho de CU deve ser executada pela AME.

Os modelos são aplicados com a premissa de evitar que determinados tamanhos de CU não sejam processados pela AME, quando provavelmente a AME não seria selecionada como o melhor modo de predição para esse tamanho de CU.

7.4 Resultados

A implementação permite a avaliação experimental da redução do tempo de codificação e perda de eficiência na qualidade de imagem para a proposta apresentada. A metodologia utilizada para análise dos resultados foi descrita no Capítulo 5.

O software utilizado para os experimentos é VTM na versão 16.2 (BROWNE; YE; KIM, 2022), executado em um servidor possuindo o sistema operacional 20.04.4 LTS, processador Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz, com 8 núcleos e 32GB RAM. Para a análise de resultados, diferentemente da mineração de dados, foram utilizadas as sequências de vídeo indicadas na CTC, permitindo uma maior imparcialidade nos resultados. Para o experimento foi utilizado as sequência de vídeo considerando as classes A1, A2, B, C e D, para os primeiros 40 quadros de cada sequência de vídeo.

A Tabela 12 apresenta os resultados do método proposto, considerando a redução do tempo total de codificação (**TR**), a redução do tempo da AME (**TR AME**). A

eficiência de codificação é apresentada em termos de *Bjontegaard Delta-bitrate* (**BD-BR**) (BJONTEGAARD, 2001).

Tabela 12 – Resultados da redução de tempo e eficiência de codificação.

Classe	**Vídeo**	**TR**	**TR AME**	**BD-BR**
A1/A2	Tango2	2,27%	16,40%	0,14%
	FoodMarket4	3,04%	16,07%	0,04%
	Campfire	1,93%	25,19%	0,01%
	CatRobot	5,01%	24,00%	0,17%
	DaylightRoad2	3,01%	17,16%	0,21%
	ParkRunning3	2,35%	11,59%	0,09%
	Média	2,94%	18,40%	0,11%
B	MarketPlace	0,16%	8,26%	-0,01%
	RitualDance	-0,44%	11,71%	0,01%
	Cactus	3,32%	19,79%	0,12%
	BasketballDrive	2,13%	11,57%	0,06%
	BQTerrace	4,88%	37,27%	0,18%
	Média	2,01%	17,72%	0,07%
C	BasketballDrill	3,79%	26,64%	-0,08%
	BQMall	3,26%	27,58%	0,04%
	PartyScene	4,60%	25,23%	0,29%
	RaceHorsesC	4,94%	13,32%	-0,02%
	Média	4,15%	23,19%	0,06%
D	BasketballPass	2,77%	18,97%	0,09%
	BQSquare	3,94%	35,81%	0,15%
	BlowingBubbles	2,44%	16,38%	-0,01%
	RaceHorses	1,46%	10,30%	-0,18%
	Média	2,65%	20,37%	0,01%
	Média Geral	**2,89%**	**19,64%**	**0,07%**

Fonte: Elaborado pelo autor.

A solução proposta é capaz de reduzir, em média, 2,89% do tempo total de codificação e 19,64% o tempo total de processamento da AME, com uma insignificante perda de qualidade de 0,07% no BD-BR.

O média de redução de tempo total do codificador e da AME, possuem comportamento similar para todas as classes de vídeo avaliadas, o que significa que a proposta é estável para ser usada com diferentes resoluções de vídeo. Em termos de eficiência de codificação, os resultados médios de todas as classes também estão próximos da média global.

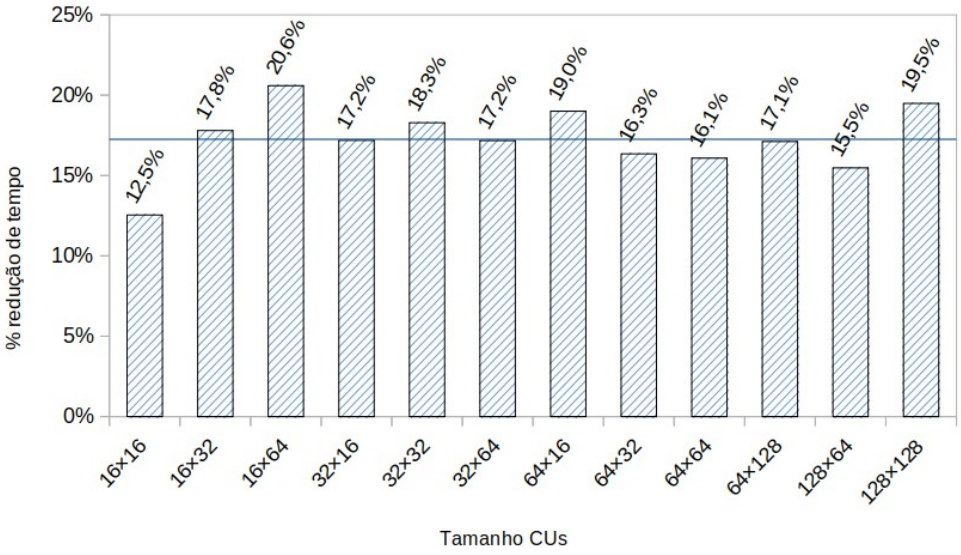
Considerando as sequências de vídeo, o melhor resultado obtido foi com *CatRobot* com 5,01% de redução do tempo total do codificador, e o pior foi encontrado em *RitualDance*, onde foi notado um pequeno aumento no tempo. *BQTerrace* e *BQSquare* obtiveram o melhor percentual de redução de tempo da AME, 37,27% e 35,81% respectivamente. Por

outro lado, o pior resultado foi encontrado em *MarketPlace*, com uma redução de 8,26% de redução de tempo da AME.

Quando considerados os resultados de eficiência de codificação, em quatro casos (valores negativos), a solução proposta alcança um pequeno ganho em BD-BR, e o melhor caso é encontrado em *RaceHorses*, com -0,18% em BD-BR. O prior resultado é encontrado em *PartyScene* com uma perda de 0,29% em BD-BR. A principal conclusão é que o método proposto alcança um mínimo ou quase nenhum impacto em termos de eficiência de codificação.

Na Figura 20 é apresentado o percentual de redução do tempo de processamento da AME para cada tamanho de CU, obtido pelo método proposto. As reduções alcançadas variam entre 12,5% na CUs 16×16 e 20,6% nas CUs 16×64, variando consistentemente em torno da média.

Figura 20 – Gráfico redução tempo AME por tamanho de CU.



Fonte: Próprio Autor.

A comparação entre o método proposto e os trabalhos relacionados pode ser vista na Tabela 13, porém ela não é completamente equivalente, uma vez que os trabalhos relacionados estão usando diferentes versões do VTM: 3.0 em (PARK; KANG, 2019), 9.0 em (DUARTE et al., 2022) e 10.0 em (JUNG; JUN, 2021). Sendo que novas melhorias são implementadas a cada nova versão do VTM, alterando a maneira de codificação durante a execução, isso pode influenciar na comparação entre os resultados. Já o trabalho (JUNG; JUN, 2021) não utiliza as mesmas sequencias de vídeo utilizadas em nosso trabalho para avaliar seus resultados, o que não permite realizar a comparação entre os trabalhos.

Considerando os dois trabalhos que utilizam as mesmas sequências de vídeos que nosso trabalho, (DUARTE et al., 2022) e (PARK; KANG, 2019), mesmo esses trabalhos usando versões muito mais antigas do VTM, nossa solução possui o menor impacto em

Tabela 13 – Comparação com trabalhos relacionados.

Trabalhos	VTM Versão	TR	TR AME	BD-BR	TR/BD-BR
Nosso	16,2	3%	20%	0,07%	43
Duarte	9,0	8%	44%	0,18%	44
Park	3,0	5%	37%	0,1%	50

termos de eficiências de codificação "BD-BR" (0,07% contra 0,18% e 0,1% respectivamente). Por outro lado, nosso trabalho apresenta a menor redução em termos de redução de tempo de codificação da AME "TR AME" (20% sobre 44% e 37% respectivamente). "TR/BD-BR", que é utilizado para avaliar a economia de tempo obtida para cada 1% de perda de eficiência de codificação, nesse ponto nosso trabalho apresenta praticamente o mesmo resultado de (DUARTE et al., 2022). Mas, novamente, como as versões do VTM são diferentes, não é possível garantir que os métodos anteriores alcançarão bons resultados na versão atual do VTM.

Outra importante consideração é que nossa proposta aplica a decisão de executar ou não a AME para determinado tamanho de CUs, em somente uma etapa. Os trabalhos (PARK; KANG, 2019) e (DUARTE et al., 2022), aplicam essa decisão em duas etapas. Portanto, nosso trabalho poderia explorar mais etapas do AME para melhorar seus resultados. Nesse caso dois novos modelos podem ser usados para definir se a *Affine* com 4-parâmetros e com 6-parâmetros deve ser ignorada ou não. Caso a CU não seja ignorada no primeiro modelo RFC, então a nova solução pode ser aplicada para definir se cada tamanho de CU deve ser ignorado ou não dentro de cada um dos modelos *Affine*. Esta solução multinível tende a melhorar as reduções de tempo ganhos.

7.5 Conclusão

Esse trabalho propõe um método para acelerar a codificação do VVC focando na etapa AME, utilizando técnicas de Aprendizado de Máquina com o método e classificação Florestas Aleatórias. Para cada tamanho de CU, um modelo foi criado para decidir se a AME deve ser aplicada ou não para o tamanho da CU. Os resultados experimentais demonstram que a solução proposta alcança em média uma redução de tempo de codificação da AME de 20%, a redução média no tempo total de codificação fica em torno de 3%, com uma insignificante perda de qualidade de imagem de 0,07% em BD-BR. Quando comparados com o estado da arte, os resultados alcançados são muito competitivos, mesmo comparados com trabalhos relacionados rodando sobre versões mais antigas do VTM, evitando assim uma comparação equivalente.

8 CONCLUSÃO E TRABALHOS FUTUROS

O aumento significativo do volume de dados de vídeo exige dos codificadores taxas de compressão cada vez maiores mantendo a qualidade de imagem. Atualmente o VVC é o codificador de vídeo estado-da-arte com a proposta de entregar uma taxa de compressão de dados 50% maior que seu antecessor o HEVC. Para alcançar esse ganho de eficiência de compressão, várias ferramentas foram melhoras e outras foram incluídas. Porém esse ganho de eficiência trouxe junto um elevado custo computacional, tornando o VVC 10 vezes mais complexo que o padrão anterior, o que atualmente inviabiliza sua utilização em larga escala.

Uma das novidades incluídas no VVC foi uma ferramenta para codificar movimentos não-translacionais com maior eficiência que os modelos translacionais, chamada Estimação de Movimento *Affine* (AME). Apesar da ferramenta contribuir com a redução da taxa de dados, ela apresenta um custo computacional maior, podendo representar até 54% do tempo total da etapa de estimação de movimento.

Com o objetivo de reduzir o esforço computacional do VVC essa dissertação apresentou duas soluções com o foco na etapa de AME. Uma proposta baseada em hardware possibilitando configurar os níveis de economia de energia e outra baseada em otimização de software utilizando Aprendizado de Máquina.

A proposta baseada em hardware apresentada uma heurística configurável e de fácil implementação em hardware visando redução do consumo de energia da pedição inter-quadros do VVC. A heurística foi definida usando três pontos de operação e foi especializada para três resoluções de vídeos nas etapas da ME Unidirecional, Bidirecional, e *Affine*. Os resultados dos experimentos mostraram reduções do consumo de energia estimado entre 7,69% e 30,77%, com uma perda de qualidade de imagem variando entra 0,04% e 1%, dependendo do ponto de operação e da resolução do vídeo. Estes resultados são expressivos, principalmente ao considerar que a maioria dos aplicativos de vídeo estão sendo executados em dispositivos alimentados por bateria hoje em dia. Os experimentos também mostraram reduções de tempo variando consistentemente com os pontos de operação.

De acordo com o conhecimento do autor, este é o primeiro trabalho na literatura que apresenta uma solução configurável e de fácil implementação em hardware abordando a redução do consumo de energia da predição inter-quadros VVC. Embora alguns trabalhos relacionados se concentrem em reduções de tempo de codificação para ferramentas individuais, eles são baseados apenas em otimizações de software, que não são aplicadas diretamente a projetos de hardware.

A proposta baseada em otimização de software propõe um método para acelerar a codificação do VVC focando na etapa AME, utilizando técnicas de Aprendizado de Máquina com o método e Classificação Florestas Aleatórias (RFC). Nessa proposta para cada tamanho de CU, um modelo foi criado para decidir se a AME deve ser aplicada ou

não para um determinado tamanho da CU. O codificador foi analisado para a extração das *features* que mais contribuiriam para a precisão do modelo. O método proposto foi implementado no VTM 16.2 e avaliado baseado nas indicações da CTC, com a configuração *Random Access* habilitada. Os resultados experimentais demonstram que a solução proposta alcança em média uma redução de tempo de codificação da AME de 20%, a redução média no tempo total de codificação fica em torno de 3%, com uma insignificante perda de qualidade de imagem de 0,07% em BD-BR.

Em ambas as soluções foi alcançado os resultados esperados, reduzindo a complexidade de codificação do VVC com perdas muito baixas de qualidade de imagem.

Ainda existem diversas possibilidade a serem exploradas para reduzir a complexidade do VVC, então como propostas de trabalhos futuros, para a solução em hardware, a heurística poderia ser estendida para outros pontos de controle, permitindo diferentes níveis de economia de energia ou até mesmo ser aplicada em outras etapas do codificador diferentes da ME. Quanto a proposta para otimização de software, seria desenvolver uma classificação multi-nível, onde além da apresentada no trabalho, acrescentar modelos RFC para decidir se o modelo AME de 6-parâmetros deve ser executado para determinado tamanho de CU ou não, ou ainda desenvolver modelos de RFC específicos para cada resolução de vídeo, deixando os modelos ainda mais precisos.

REFERÊNCIAS

A. CHEN J., Y. Y. K. S. B. *Algorithm description for Versatile Video Coding and Test Model 14 (VTM 14)*. 2021. Joint Video Experts Team (JVET)of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 2923rd Meeting, by teleconference, 7–16 July 2021. Citado na página 57.

AMESTOY, T. et al. Tunable VVC frame partitioning based on lightweight machine learning. *IEEE Transactions on Image Processing*, v. 29, p. 1313–1328, 2020. DOI 10.1109/TIP.2019.2938670. Citado 2 vezes nas páginas 40 e 47.

BJONTEGAARD, G. Calculation of average psnr differences between rd-curves. *Proceedings of the ITU-T Video Coding Experts Group (VCEG) Thirteenth Meeting*, 01 2001. Citado 3 vezes nas páginas 52, 58 e 70.

BOSSEN, F. et al. *VTM common test conditions and software reference configurations for SDR video*. 2020. Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29. Citado 2 vezes nas páginas 51 e 62.

BRADSHAW, D.; KINGSBURY, N. Combined affine and translational motion compensation scheme using triangular tessellations. In: *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*. [S.l.: s.n.], 1997. v. 4, p. 2645–2648 vol.4. DOI - 10.1109/ICASSP.1997.595332. Citado 2 vezes nas páginas 33 e 46.

BRANDENBURG, J. et al. Towards fast and efficient VVC encoding. In: *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*. [S.l.]: IEEE, 2020. Citado na página 16.

BREIMAN, L. Random forests. *Machine Learning*, Springer Science and Business Media LLC, v. 45, n. 1, p. 5–32, 2001. DOI - 10.1023/a:1010933404324. Citado 2 vezes nas páginas 39 e 43.

BROSS, B. et al. Developments in international video coding standardization after AVC, with an overview of versatile video coding (VVC). *Proceedings of the IEEE*, Institute of Electrical and Electronics Engineers (IEEE), v. 109, n. 9, p. 1463–1493, sep 2021. DOI - 10.1109/jproc.2020.3043399. Citado 3 vezes nas páginas 16, 26 e 35.

BROSS, B. et al. Overview of the versatile video coding (VVC) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology*, Institute of Electrical and Electronics Engineers (IEEE), v. 31, n. 10, p. 3736–3764, oct 2021. DOI - 10.1109/tcsvt.2021.3101953. Citado 5 vezes nas páginas 15, 16, 26, 36 e 54.

BROWNE, A.; YE, Y.; KIM, S. H. *Algorithm description for Versatile Video Coding and Test Model 16 (VTM16)*. [S.l.]: Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29, 2022. WG 05 MPEG Joint Video Coding Team(s) with ITU-T SG 16. Citado na página 69.

CHAKRABARTI, I.; BATTA, K. N. S.; K., C. S. Introduction. In: *Motion Estimation for Video Coding*. [S.l.]: Springer International Publishing, 2015. p. 1–10. DOI - 10.1007/978-3-319-14376-7_1.$Citadonap\acute{a}gina19$.

CHEN, M. et al. Efficient partition decision based on visual perception and machine learning for H.266/versatile video coding. *IEEE Access*, v. 10, p. 42141–42150, 2022. DOI 10.1109/ACCESS.2022.3168155. Citado 2 vezes nas páginas 40 e 48.

CHIEN, W. et al. Motion vector coding and block merging in the versatile video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, Institute of Electrical and Electronics Engineers (IEEE), v. 31, n. 10, p. 3848–3861, oct 2021. DOI - 10.1109/tcsvt.2021.3101212. Citado na página 33.

CHINNAMGARI, S. K. *R Machine Learning Projects*. Packt Publishing, 2019. ISBN 1789807948. Disponível em: <https://www.ebook.de/de/product/35298520/sunil_kumar_chinnamgari_r_machine_learning_projects.html>. Citado na página 39.

CISCO. *Cisco Visual Networking Index: Forecast and Trends, 2017–2022*. 2019. <https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.pdf>. Acessado em: 04/12/2021. Citado na página 15.

CISCO. *Cisco Annual Internet Report (2018–2023)*. 2020. <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html>. Acessado em: 04/12/2021. Citado na página 15.

CUTLER, A.; CUTLER, D. R.; STEVENS, J. R. Random forests. In: *Ensemble Machine Learning*. [S.l.]: Springer US, 2012. p. 157–175. DOI - 10.1007/978-1-4419-9326-7$_5$. *Citadonapágina*43.

DAEDE, T.; NORKIN, A.; BRAILOVKKIY, I. *Videocodec testing and quality measurement.* 2018. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-netvc-testing-07. Citado na página 62.

DUARTE, A. et al. Fast affine motion estimation for vvc using machine-learning-based early search termination. In: *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*. [S.l.: s.n.], 2022. p. 1–5. DOI - 10.1109/ISCAS48785.2022.9937973. Citado 4 vezes nas páginas 47, 59, 71 e 72.

GONZALEZ, R. C.; WOODS, R. E. *Processamento de imagens digitais*. [S.l.]: Editora Blucher, 2009. Citado na página 14.

GONÇALVES, P. H. R. *Um Esquema Rápido Baseado em Aprendizado de Máquina para a Predição Interquadros do Codificador de Vídeo VVC*. Dissertação (mathesis) — Universidade Federal de Pelotas., Abril 2021. Citado 2 vezes nas páginas 29 e 31.

HARRINGTON, P. *Machine Learning in Action*. MANNING PUBN, 2012. ISBN 1617290181. Disponível em: <https://www.ebook.de/de/product/15619827/peter_harrington_machine_learning_in_action.html>. Citado na página 42.

HE, Q. et al. Random forest based fast CU partition for VVC intra coding. In: *2021 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. [S.l.: s.n.], 2021. p. 1–4. DOI 10.1109/BMSB53066.2021.9547117. Citado 2 vezes nas páginas 40 e 48.

HOANG, D.; LONG, P.; VITTER, J. Efficient cost measures for motion estimation at low bit rates. *IEEE Transactions on Circuits and Systems for Video Technology*, Institute of Electrical and Electronics Engineers (IEEE), v. 8, n. 4, p. 488–500, 1998. DOI - 10.1109/76.709413. Citado na página 32.

HUANG, Y. et al. Block partitioning structure in the VVC standard. *IEEE Transactions on Circuits and Systems for Video Technology*, Institute of Electrical and Electronics Engineers (IEEE), v. 31, n. 10, p. 3818–3833, oct 2021. DOI - 10.1109/tcsvt.2021.3088134. Citado 3 vezes nas páginas 28, 29 e 30.

JUNG, S.; JUN, D. Context-based inter mode decision method for fast affine prediction in versatile video coding. *Electronics*, MDPI AG, v. 10, n. 11, p. 1243, may 2021. DOI - 10.3390/electronics10111243. Citado 4 vezes nas páginas 46, 59, 63 e 71.

KIM, T. S. et al. Fast integer motion estimation with bottom-up motion vector prediction for an HEVC encoder. *IEEE Transactions on Circuits and Systems for Video Technology*, Institute of Electrical and Electronics Engineers (IEEE), v. 28, n. 12, p. 3398–3411, dec 2018. DOI - 10.1109/tcsvt.2017.2759245. Citado na página 33.

LEMAÎTRE, G.; NOGUEIRA, F.; ARIDAS, C. K. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *The Journal of Machine Learning Research*, JMLR. org, v. 18, n. 1, p. 559–563, 2017. Citado na página 64.

LI, J.; ZHANG, S.; YANG, F. Random forest accelerated cu partition for inter prediction in H.266/VVC. In: *2022 IEEE International Conference on Multimedia and Expo (ICME)*. [S.l.: s.n.], 2022. p. 01–06. DOI 10.1109/ICME52920.2022.9859664. Citado 2 vezes nas páginas 40 e 48.

LIU, H. et al. Adaptive motion vector resolution for affine-inter mode coding. In: *2019 Picture Coding Symposium (PCS)*. [S.l.]: IEEE, 2019. DOI - 10.1109/pcs48520.2019.8954531. Citado na página 36.

LOOSE, M. et al. A hardware-friendly and configurable heuristic targeting VVC inter-frame prediction. In: *2022 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. [S.l.: s.n.], 2022. Citado na página 53.

MAIMON, O.; ROKACH, L. (Ed.). *Data Mining and Knowledge Discovery Handbook*. [S.l.]: Springer US, 2010. DOI - 10.1007/978-0-387-09823-4. Citado 2 vezes nas páginas 40 e 42.

MARPE, D.; WIEGAND, T.; SULLIVAN, G. The h.264/mpeg4 advanced video coding standard and its applications. *IEEE Communications Magazine*, v. 44, n. 8, p. 134–143, 2006. DOI - 10.1109/MCOM.2006.1678121. Citado na página 26.

MERCAT, A.; VIITANEN, M.; VANNE, J. UVG dataset. In: *Proceedings of the 11th ACM Multimedia Systems Conference*. [S.l.]: ACM, 2020. p. 297–302. DOI 10.1145/3339825.3394937. Citado na página 62.

PAKDAMAN, F. et al. Complexity analysis of next-generation VVC encoding and decoding. In: *2020 IEEE International Conference on Image Processing (ICIP)*. [S.l.]: IEEE, 2020. DOI - 10.1109/icip40778.2020.9190983. Citado na página 27.

PAPAGEORGIOU, E.; STYLIOS, C.; GROUMPOS, P. A combined fuzzy cognitive map and decision trees model for medical decision making. In: *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*. [S.l.: s.n.], 2006. p. 6117–6120. DOI - 10.1109/IEMBS.2006.260354. Citado na página 40.

PARK, S.; KANG, J. Fast affine motion estimation for versatile video coding (VVC) encoding. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 7, p. 158075–158084, 2019. DOI - 10.1109/access.2019.2950388. Citado 6 vezes nas páginas 17, 46, 50, 59, 71 e 72.

PARMAR, A.; KATARIYA, R.; PATEL, V. A review on random forest: An ensemble classifier. In: *International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI) 2018*. [S.l.]: Springer International Publishing, 2018. p. 758–763. Doi - 10.1007/978-3-030-03146-6$_8$6.*Citadonapágina*43.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. *The Journal of machine Learning research*, JMLR. org, v. 12, p. 2825–2830, 2011. Citado na página 67.

RICHARDSON, I. E. G. *Video Codec Design*. John Wiley Sons, 2002. ISBN 0471485535. Disponível em: <https://www.ebook.de/de/product/3056150/richardson_video_codec_design.html>. Citado 5 vezes nas páginas 19, 20, 21, 23 e 24.

SALDANHA, M. et al. Configurable fast block partitioning for VVC Intra coding using light gradient boosting machine. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 32, n. 6, p. 3947–3960, 2022. DOI 10.1109/TCSVT.2021.3108671. Citado 2 vezes nas páginas 40 e 48.

SALDANHA, M. et al. Fast transform decision scheme for vvc intra-frame prediction using decision trees. In: *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*. [S.l.: s.n.], 2022. p. 1948–1952. DOI 10.1109/ISCAS48785.2022.9938000. Citado 2 vezes nas páginas 40 e 48.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, v. 3, n. 3, p. 210–229, 1959. DOI - 10.1147/rd.33.0210. Citado na página 39.

SANDVINE. *The Global Internet Phenomena Report COVID-19 Spotlight*. 2020. <https://www.sandvine.com/phenomena>. Acessado em: 06/12/2021. Citado na página 15.

SANDVINE. *The Mobile Internet Phenomena Report*. 2021. <https://www.sandvine.com/phenomena>. Acessado em: 06/12/2021. Citado na página 15.

SIGNATURE, D. *Diferença entre resoluções hd, full hd, ultra hd, 2K, 4k, 8k, 10K*. 2022. <https://dsplay.tv/site/suporte/qual-a-diferenca-entre-resolucoes-hd-full-hd-ultra-hd-2k-4k-8k-10k/>. Acessado em: 09/03/2022. Citado na página 16.

SIQUEIRA, I.; CORREA, G.; GRELLERT, M. Rate-distortion and complexity comparison of HEVC and VVC video encoders. In: *2020 IEEE 11th Latin American Symposium on Circuits & Systems (LASCAS)*. [S.l.]: IEEE, 2020. DOI - 10.1109/lascas45839.2020.9069036. Citado 5 vezes nas páginas 16, 17, 26, 32 e 49.

SULLIVAN, G. J. et al. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 22, n. 12, p. 1649–1668, 2012. DOI - 10.1109/TCSVT.2012.2221191. Citado na página 15.

SUTHAHARAN, S. Decision tree learning. In: *Machine Learning Models and Algorithms for Big Data Classification*. [S.l.]: Springer US, 2016. p. 237–269. DOI - 10.1007/978-1-4899-7641-3_10. *Citadonapágina*42.

SZE, V.; BUDAGAVI, M.; SULLIVAN, G. J. (Ed.). *High Efficiency Video Coding (HEVC)*. [S.l.]: Springer International Publishing, 2014. DOI - 10.1007/978-3-319-06895-4. Citado 2 vezes nas páginas 21 e 26.

VEERAVALLI, A. et al. Covariance analysis of non-translational motion-compensated frame differences. In: *Proceedings of the Thirty-Seventh Southeastern Symposium on System Theory (SSST05)*. Los Alamitos, CA, USA: IEEE Computer Society, 2005. p. 462–465. DOI - 10.1109/SSST.2005.1460958. Citado 2 vezes nas páginas 33 e 46.

VTM. *VVC VTM reference software*. 2022. <https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM>. Acessado em: 10/03/2022. Citado 2 vezes nas páginas 26 e 51.

YANG, H. et al. Subblock-based motion derivation and inter prediction refinement in the versatile video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, Institute of Electrical and Electronics Engineers (IEEE), v. 31, n. 10, p. 3862–3877, oct 2021. DOI - 10.1109/tcsvt.2021.3100744. Citado na página 37.

ZHANG, K. et al. An improved framework of affine motion compensation in video coding. *IEEE Transactions on Image Processing*, Institute of Electrical and Electronics Engineers (IEEE), v. 28, n. 3, p. 1456–1469, mar 2019. DOI - 10.1109/tip.2018.2877355. Citado 3 vezes nas páginas 33, 34 e 46.

ZHANG, Q. et al. Fast CU partition and intra mode decision method for H.266/VVC. *IEEE Access*, v. 8, p. 117539–117550, 2020. DOI 10.1109/ACCESS.2020.3004580. Citado 2 vezes nas páginas 40 e 47.

ZHANG, T.; MAO, S. An overview of emerging video coding standards. *GetMobile: Mobile Computing and Communications*, Association for Computing Machinery (ACM), v. 22, n. 4, p. 13–20, may 2019. DOI - 0.1145/3325867.3325873. Citado 2 vezes nas páginas 14 e 34.