
Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Departamento de Engenharia Elétrica
Programa de Pós Graduação em Engenharia Elétrica

Tese de Doutorado

# Técnica de Aprendizagem Automática Aplicada a um Codificador HEVC em Tempo Real

Jean Felipe Fonseca de Oliveira

Campina Grande – PB
Fevereiro de 2016

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Departamento de Engenharia Elétrica
Programa de Pós Graduação em Engenharia Elétrica

# Técnica de Aprendizagem Automática Aplicada a um Codificador HEVC em Tempo Real

Jean Felipe Fonseca de Oliveira

Tese de Doutorado apresentada à Coordenação do Programa de Pós Graduação em Engenharia Elétrica do Departamento de Engenharia Elétrica da Universidade Federal de Campina Grande como requisito necessário para obtenção do grau de Doutor em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Comunicações.

Marcelo Sampaio de Alencar
Orientador

Campina Grande – PB, Fevereiro de 2016

**FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG**

O48t Oliveira, Jean Felipe Fonseca de.
Técnica de aprendizagem automática aplicada a um codificador HEVC em tempo real / Jean Felipe Fonseca de Oliveira. – Campina Grande, 2016.
94f. : il. color.

Tese (Doutorado em Engenharia Elétrica) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.
"Orientação: Prof. Marcelo Sampaio de Alencar".

1. Engenharia Elétrica. 2. Aprendizado Automático. 3. Codificador HEVC. 4. Aprendizado *Online*. I. Alencar, Marcelo Sampaio de. II. Título.

CDU 621.3(043)

**"TÉCNICA DE APRENDIZAGEM AUTOMÁTICA APLICADA A UM CODIFICADOR HEVC EM TEMPO REAL"**

**JEAN FELIPE FONSECA DE OLIVEIRA**

**TESE APROVADA EM 13/05/2016**

**MARCELO SAMPAIO DE ALENCAR, Ph.D., UFCG**
**Orientador(a)**

**FRANCISCO MADEIRO BERNARDINO JÚNIOR, D.Sc, UPE**
**Examinador(a)**

**WASLON TERLLIZZIE ARAÚJO LOPES, D.Sc., UFPB**
**Examinador(a)**

**JOSÉ EWERTON POMBO DE FARIAS, Dr., UFCG**
**Examinador(a)**

**MAX HENRIQUE MACHADO COSTA, Ph.D., UNICAMP**
**Examinador(a)**

**CAMPINA GRANDE - PB**

*Aos meus pais, João Batista e Maria Gorette, que ladrilharam todo esse meu caminho.*
*À minha esposa, Danielle e minhas filhas, Marina e Letícia, que são as luzes dos meus olhos.*

*Não é necessário que um homem seja ativamente mau, a fim de fazer fracassos na vida; a simples inação irá realizá-los. A natureza tem em todos os lugares por escrito o seu protesto contra a ociosidade; tudo o que deixa de lutar, que permanece inativo, se deteriora rapidamente. É a luta em direção a um ideal, o esforço constante para crescer sempre, que desenvolve maturidade e caráter.*
*James Terry White*

# Agradecimentos

À Deus.

À minha amada esposa Danielle Menezes Dantas de Oliveira, principal apoio dessa jornada. Seu apoio, paciência e amor foram imprescindíveis para finalizar esse trabalho. Você e nossas pequenas, Marina e Letícia, foram sempre a minha motivação. Nada é mais importante que seus sorrisos.

Aos meus pais, João Batista e Maria Gorette, meus maiores mestres, por seu amor incondicional e suporte em todas as etapas da minha vida. Seus conselhos foram sempre palavras de Deus para mim. Honrarei vocês por toda minha vida. Seus exemplos de vida foram minha maior inspiração.

Às minhas irmãs Jeanne e Jeanine, que sempre me apoiaram em tudo que fiz. Amo vocês duas.

À toda minha família pelo apoio e confiança que sempre me dedicaram. Em especial os meus avós, Ana Ninô(*in memoriam*), Domerina e Severino(*in memoriam*). Há muito deles nessa história.

Aos meus sogros Wellington e Ângela, meu agradecimento vai muito além do que palavras. Será uma atitude constante. Extendo esse sentimento à toda família, em especial às pequenas Sofia e Cecília, que alegram nossos dias.

Ao professor Marcelo Sampaio de Alencar, pela confiança, conhecimentos cedidos e pela amizade. Meus mais profundos agradecimentos. Foi um prazer e uma honra conviver com um verdadeiro cientista durante esses últimos dez anos.

À TPV *Technology Limited*, que entendendo a importância deste trabalho para minha formação, forneceu o apoio necessário. Registro meus agradecimentos à toda equipe, em especial para os amigos Denis Hipólito de Araújo e Rômulo Fabrício.

Aos amigos que contribuíram de alguma forma, meus sinceros agradecimentos. Em especial aos Professores Paulo Ribeiro Lins Jr. e Carlos Danilo Miranda Régis.

# Resumo

O padrão HEVC (*High Efficiency Video Coding*) é o mais recente padrão para codificação de vídeos e tem uma complexidade computacional muito maior do que seu antecessor, o padrão H.264. A grande eficiência de codificação atingida pelo codificador HEVC é obtida com um custo computacional bastante elevado. Esta tese aborda oportunidades de reduzir essa carga computacional. Dessa forma, um algoritmo de decisão prematura de divisão de uma unidade de codificação é proposto para o codificador HEVC, terminando prematuramente o processo de busca pelo melhor particionamento baseado em um modelo de classificação adaptativo, criado em tempo de execução. Esse modelo é gerado por um processo de aprendizado *online* baseado no algoritmo Pegasos, que é uma implementação que aplica a resolução do gradiente estocástico ao algoritmo SVM (*Support Vector Machine*). O método proposto foi implementado e integrado ao codificador de referência HM 16.7. Os resultados experimentais mostraram que o codificador modificado reduziu o custo computacional do processo de codificação em até 50%, em alguns casos, e aproximadamente 30% em média, com perdas de qualidade desprezíveis para os usuários. De modo geral, esse processo resulta em reduzidas perdas de qualidade, no entanto, alguns resultados mostraram pequenos ganhos em eficiência de compressão quando comparados com os resultados do codificador HM 16.7.

**Palavras-chave:** HEVC, Aprendizado Automático, Máquina de Vetores Suportes, Seleção de Atributos, Aprendizado Online, Otimização

# Abstract

The most recent video coding standard, the High Efficiency Video Coding (HEVC), has a higher encoding complexity when compared with H.264/AVC, which means a higher computational cost. This thesis presents a review of the recent literature and proposes an algorithm that reduces such complexity. Therefore, a fast CU (Coding Unit) splitting algorithm is proposed for the HEVC encoder, which terminates the CU partitioning process at an early phase, based on an adaptive classification model. This model is generated by an online learning method based on the Primal Estimated sub-GrAdient SOlver for SVM (Pegasos) algorithm. The proposed method is implemented and integrated in the HEVC reference source code on its version 16.7. Experimental results show that the proposed method reduces the computational complexity of the HEVC encoder, up to 50% in some cases, with negligible losses, and shows an average computational reduction of 30%. This process results in reduced coding efficiency losses, however, some results showed a nearby 1% of BD-Rate (Bjontegaard Delta) gains in the Low Delay B configuration, without using an offline training phase.

**Keywords:** HEVC, Machine Learning, SVM, Feature Selection, Online Learning, Video coding optimization

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Lista de Siglas

| | |
|---|---|
| ABNT | Associação Brasileira de Normas Técnicas |
| AMVP | *Advanced motion vector prediction* |
| AVC | *Advanced Video Coding* |
| CABAC | *Context-Based Arithmetic Coding* |
| CAVLC | *Context-Adaptive Variable-Length Codes* |
| CTB | *Coding Tree Block* |
| CTU | *Coding Tree Unit* |
| CU | *Coding Unit* |
| DCT | *Discrete Cosine Transform* |
| DVB-T | *Digital Video Broadcasting - Terrestrial* |
| EO | *Edge Offset* |
| EPZS | *Enhanced Predictive Zonal Search* |
| FRExt | *Fidelity Range Extensions* |
| HEVC | *High Efficiency Video Coding* |
| IDFT | *Inverse Discrete Fourier Transform* |
| IGAE | *Information Gain Attribute Evaluation* |
| JCT-VC | *Joint Collaborative Team on Video Coding* |
| ISDB-T | *Integrated Services Digital Broadcasting Terrestrial* |
| ISO | *International Organization for Standardization* |
| IEC | *International Electrotechnical Commission* |
| ITU-T | *International Telecommunication Union - Telecommunication Standardization Sector* |
| MPEG | *Motion Picture Experts Group* |
| NAL | *Network Abstraction Layer* |
| PSNR | *Peak Signal-to-Noite Ratio* |
| PU | *Prediction Block* |
| PU | *Prediction Unit* |
| SAO | *Sample Adaptative Offset* |
| SI | *Spatial Information* |
| SVM | *Support Vector Machine* |

| | |
|---|---|
| TI | *Temporal Information* |
| TMuC | *Test Model under Consideration* |
| TB | *Transforming Block* |
| TU | *Transforming Unit* |
| UHD | *Ultra High Definition* |
| VCEG | *Video Coding Experts Group* |

# Lista de Símbolos

| | |
|---|---|
| $\lambda$ | Fator que otimiza a relação taxa para distorção. |
| $\Omega$ | Identificação do codificador proposto nesta tese. |

# CAPÍTULO 1

# Introdução

Durante a década passada houve um significativo aumento da demanda por serviços de vídeo digital, como por exemplo, transmissões de televisão em alta definição, canais de televisão transmitidos pela Internet, vídeo-conferências e vídeo sob-demanda. Para viabilizar esses serviços, significativos avanços no campo de compressão de sinais de vídeo foram alcançados para permitir a redução do espaço de armazenameto e atender às limitações da largura de banda disponível. Dessa forma, gerou-se um apelo por novas técnicas de codificação mais poderosas do que as especificadas pelo H.264 (WANG *et al.*, 2013).

A resolução e os recursos de exibição de uma ampla gama de produtos eletrônicos, desde televisores a celulares, têm melhorado nos últimos tempos. Isso deve fomentar o desenvolvimento de tecnologias relacionadas com a televisão em ultra-alta definição (UHDTV – *Ultra High Definition Television*), com resoluções de até $7680 \times 4320$ *pixels* e o crescente aumento no tamanho e resolução das telas de dispositivos móveis, como *smartphones* e *tablets* (NIGHTINGALE *et al.*, 2012).

O fluxo contínuo de vídeo (*streaming*) é uma aplicação que requer uso intensivo da taxa disponível para transmissão e atualmente responde por um percentual significativo do tráfego de Internet. Em 2012, a Cisco relatou que o *streaming* de vídeo para dispositivos utilizados por usuários comuns transferiram 56 exabytes pela Internet em 2010 e que haveria um crescimento para 403 exabytes em 2015 (NIGHTINGALE *et al.*, 2012). Com a adoção de dispositivos de alta resolução, tanto com fio (por exemplo, televisores UHD) como sem fio (*smartphones* e *tablets*), os operadores de rede terão que enfrentar novos desafios na oferta de serviços com largura de banda suficiente para satisfazer a demanda do consumidor.

Estes novos desafios podem ser satisfeitos pela melhoria da taxa de compressão, reduzindo assim a exigência de largura de banda do padrão de codificação corrente, o H.264/AVC (*Advanced Video Coding*). O novo padrão, o *High Efficiency Video Coding* (HEVC), tem como objetivo resolver este problema, fornecendo um aumento de 50% na eficiência de compressão sobre o padrão H.264/AVC, mantendo o mesmo nível de qualidade visual percebida pelos usuários (SULLIVAN *et al.*, 2012).

O padrão de codificação de vídeo HEVC é o mais recente esforço conjunto entre as organizações de padronização ITU-T VCEG (*Video Coding Experts Group*) e a ISO/IEC MPEG (*Moving Picture Experts Group*) em uma parceria conhecida como JCT-VC (*Joint Collaborative Team on Video Coding*). A primeira edição do padrão HEVC foi publicada em janeiro de 2013, em duas normas com mesmo conteúdo divulgadas pelos orgãos envolvidos.

O HEVC foi projetado para atender a todas as aplicações existentes do H.264/MPEG-4 AVC e, principalmente, dois requisitos específicos: codificação de vídeos com alta resolução e o uso de arquiteturas de processamento parelelo (VANNE *et al.*, 2012). A sintaxe do HEVC é genérica ao ponto de possibilitar uma adequação satisfatória a outras aplicações além das duas citadas (SULLIVAN *et al.*, 2012). A Figura 1.1 ilustra a nova arquitetura desenvolvida para o codificador HEVC. O aprofundamento das novas ferramentas de codificação e novos elementos da sintaxe foram estudados nesta tese.

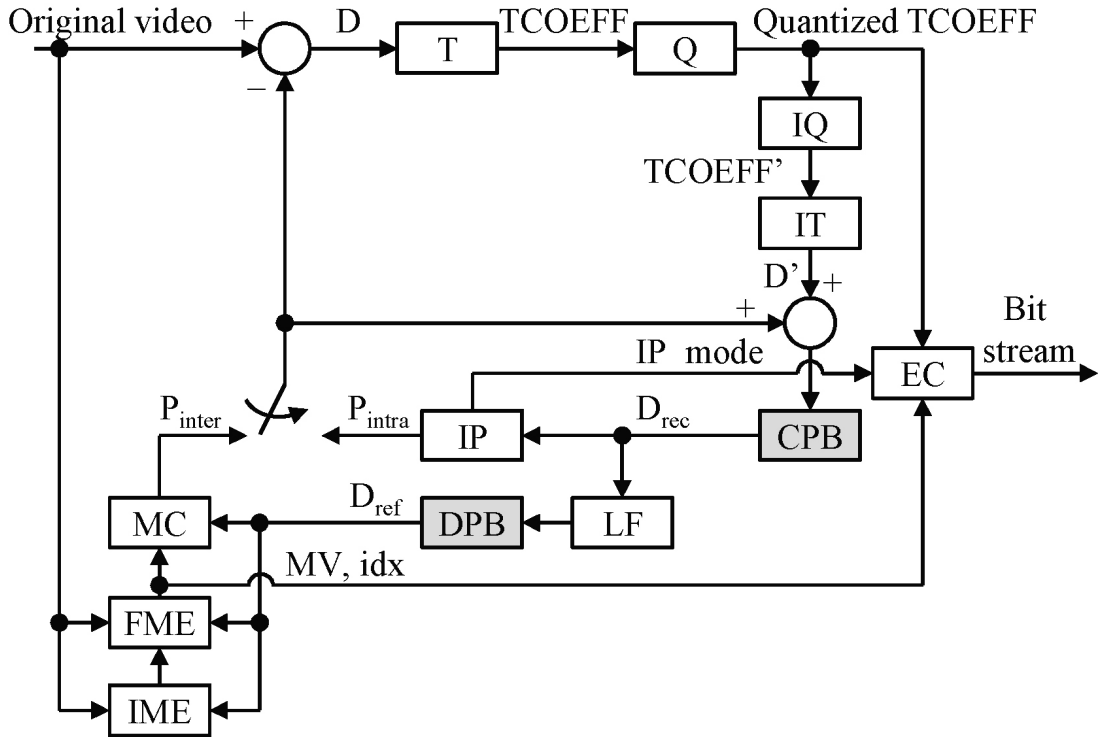



**Figura 1.1** Arquitetura de blocos do codificador HEVC.

As aplicações multimídia têm se tornado mais flexíveis e mais poderosas com a evolução da microeletrônica e o desenvolvimento de novos métodos de processamento de sinais digitais. A limitação da largura de banda do canal e os rigorosos requisitos de reprodução de vídeo em tempo real implicam que a codificação seja um processo indispensável para muitas aplicações de comunicação visual que requerem taxas de compressão muito alta. A elevada correlação temporal entre quadros adjacentes em uma sequência de vídeo, também chamada redundância temporal, deve ser identificada e, posteriormente explorada, com o intuito de eliminar a informação reduntante (KAMOLRAT *et al.*, 2009).

Com o aumento na popularidade das comunicações de vídeo, a experiência do usuário passa a ser uma das preocupações mais importantes na concepção e avaliação de sistemas multimídia (LIU *et al.*, ). Em uma cadeia de transmissão de vídeo, vários fatores influenciam a qualidade da imagem exibida. Um desses fatores é o próprio algoritmo de codificação de fonte. Como consequência da codificação com perdas, uma degradação de qualidade do vídeo pode ser observada (WOLFF *et al.*, 2006).

Recentes avanços nas tecnologias de captura e exibição de conteúdos de vídeo de alta resolução têm levado a um interesse crescente em serviços que proporcionam uma experiência de usuário ampliada. Para prestar tais serviços, o novo formato de vídeo de Ultra Alta Definição (UHD) foi definido, como um sucessor para o formato de alta definição (HD). As especificações do formato de vídeo em ultra alta definição estão especificadas como o padrão UHDTV, que foi ratificado pelo grupo de trabalho do padrão DVB (*Digital Video Broadcasting*) e pela União Europeia de Radiodifusão (UER), conforme especificado na Recomendação BT.2020 (SUGAWARA *et al.*, 2014), e compreendem resoluções espaciais de 3840×2160 e 7680×4320 amostras de luminância por quadro, resolução temporal de até 120 quadros por segundo e profundidade de até 12 *bits* por banda de informação.

As resoluções citadas são múltiplas do padrão HDTV. O espaço de cores definido para o padrão UHD é composto pela luminância (Y), crominância vermelha (Cr) e crominância azul (Cb). Tal quantidade de dados aumenta a necessidade de uma tecnologia de compressão eficiente, pois uma transmissão UHDTV gera um volume de dados até oito vezes maior do que os dados transmitidos em um canal HDTV padrão.

Estatísticas das previsões da Cisco sobre o trafégo de dados (CISCO, 2015) mostram que, em 2020, 76% do total de informações trafegando na Internet serão dados de vídeo e, principalmente, vídeo em altas resoluções, como mostrado na Figura 1.2. Para responder à necessidade, o padrão H.265/*High Efficiency Video Coding* (HEVC) foi desenvolvido em parceria pelo grupo de especialistas em codificação de vídeo da ITU (ITU/VCEG – *Video Coding Experts Group*) e pelo grup de especialistas da ISO *Moving Picture Experts Group*. O padrão HEVC supera seu antecessor, o padrão H.264/*Advanced Video Coding* (AVC) com reduções médias de 50% na taxa de *bits* para a mesma qualidade percebida. Foi medida uma ainda maior eficiência de compressão quando comparada à compressão de conteúdo UHD, principalmente devido ao uso de grandes blocos de até 64×64 amostras de luminância. No entanto, a codificação HEVC de um sinal com resolução UHD representa uma tarefa de elevada complexidade computacional. Por esta razão, métodos para diminuir a complexidade do esquema de codificação HEVC são cruciais, especialmente quando se codifica o conteúdo UHD.

Nesta tese, o desempenho da implemenitação de referência do codificador HEVC foi analisado em termos de tempo gasto durante a realização de diferentes tarefas de codificação de conteúdos de vídeo em UHD. Com base nesta análise, um novo algoritmo é apresentado para reduzir a complexidade do módulo de predição inter-quadros, que é o processo de estimação da redundância temporal.

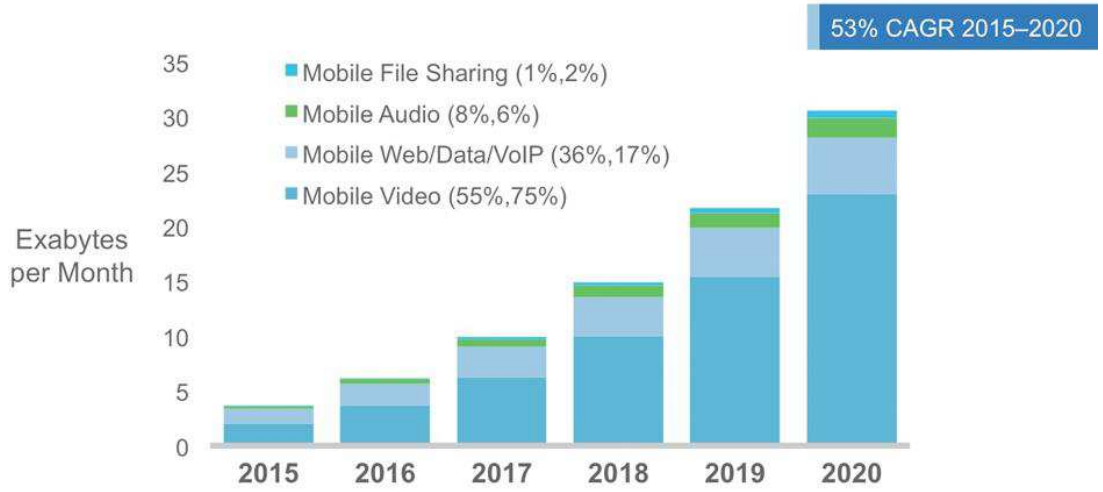



**Figura 1.2** Dados de vídeo em redes móveis representaram três quartos do total de dados móveis (CISCO, 2015).

A complexidade do HEVC torna-se um problema ainda mais crítico quando é aplicado para compressão de conteúdo em ultra-alta definição (UHD). Como resultado, o conteúdo típico UHD disponível hoje é caracterizado por uma falta inerente de detalhes finos e outras características peculiares. É razoável supor que essas características podem influenciar o desempenho da codificação, devido ao seu impacto sobre a qualidade dos blocos de referência utilizados para a etapas da predição inter e também da quantidade de resíduos não nulos resultantes após a quantização. Esses fatores podem aumentar a complexidade e prejudicar a eficiência do HEVC e, embora tenha sido projetado para lidar com compressão de sinais de grandes resoluções, existem poucos trabalhos para tratar especificamente de codificação UHD.

Vídeos em ultra-alta definição estão se tornando mais populares por proporcionarem maior qualidade percebida e por criar uma experiência de visualização mais realista, gerando um mercado de vídeo promissor. No entanto, o volume de dados desses vídeos em UHD aumenta significantemente com o aumento da resolução e da taxa de quadros. Por exemplo, um vídeo $7680 \times 4320$ a 120 quadros por segundo com 24 *bits* por *pixel*, representa um fluxo de $11,1$ *gigabytes* por segundo de dados de vídeo bruto, o que requer um esquema de compressão altamente eficiente.

O HEVC adota avançadas técnicas de codificação, incluindo um novo algoritmo de partição do bloco de codificação, 35 modos de previsão intra, a transformada discreta de senos, sofisticados algoritmos de interpolação e filtragem, entre outros. Estas técnicas melhoraram significativamente a eficiência de compressão. No entanto, elas implicam em alto custo de complexidade computacional e, consequentemente, alto custo de *hardware*, por demandar muita capacidade de processamento, acesso à memória, espaço de armazenamento e consumo de energia, que dificultam o uso de vídeos UHD em aplicações de tempo real, tais como transmissão de TV ao vivo, vídeo para dispositivos embarcados e aplicações de vigilância.

O núcleo do processo de codificação do HEVC é a unidade de codificação em árvore (*Coding Tree Unit* – CTU), conceitualmente semelhante ao macrobloco usado em padrões de codificação de vídeo anteriores. Uma CTU pode conter apenas uma unidade de codificação, ou ser dividida em múltiplas unidades com dimensões de $8 \times 8$, $16 \times 16$, $32 \times 32$ e $64 \times 64$. Cada unidade de codificação pode ser dividida em unidades de predição menores de vários modos, inclusive nos modos SKIP e MERGE. Nesse processo, o codificador verifica $4^i$ partições,$i \in [0,3]$. Com isso uma CTU pode ter $(2^4+1)^4+1 = 83522$ diferentes combinações de tamanhos de unidades de codificação. Os resíduos resultantes das etapas de exploração da redundância temporal são processados com unidades de transformada (TU – *Transform Unit*). A configuração ótima de cada unidade de codificação, de predição e de transformada são determinadas baseado em comparações recursivas da relação taxa/distorção para cada possibilidade, em uma estratégias que todas as configurações são avaliadas para a escolha da melhor, caracterizando um processo de complexidade elevada.

O objeto de estudo dessa tese é abordar o problema de redução da complexidade computacional de um codificador de vídeo HEVC para vídeo em UHD sem implicar em perdas significativas na eficiência de compressão final em termos de distorção e taxa de *bits* resultante. A complexidade computacional é medida pelo tempo de execução do processo de codificação. Considerando que, por exemplo, algumas vezes a qualidade do vídeo é limitada pelo tempo de processamento disponível, essa redução proposta pode melhorar a qualidade da experiência dos usuários finais. Em geral, a maioria dos algoritmos de aprimoramento do processo de codificação HEVC encontrados na literatura recente, e que serão descritos no próximo capítulo, são baseados na relação taxa/distorção e nas informações espaço-temporais. Correlações dessas variáveis são utilizadas para tentar abreviar o dispendioso processo de particionamento das unidades de codificação, predição e transformação. Essa abordagem tende a limitar a generalização dessas técnicas, dificultando sua execução em situações com outros conteúdos, parâmetros ou estruturas de codificação mais complexas.

Considerando o cenário apresentado, essa tese utiliza técnicas de aprendizagem de máquina, ou aprendizagem automática, usando uma abordagem que diminui a necessidade de conhecimentos profundos das correlações dos diversos parâmetros do processo de codificação HEVC. A aprendizagem automática tem sido aplicada com sucesso em problemas de inteligência artificial, reconhecimento de padrões e processamento de sinais devido a sua capacidade de aprender a partir de grandes conjuntos de dados em situações complexas e atingir resultados aprimorados. A modificação proposta foi implementada na versão 16.7 do codificador de referência do padrão HEVC. A possibilidade de executar um algoritmo de aprendizagem automática em tempo de execução é uma das principais contribuições deste trabalho. As simulações mostraram que o classificador utilizado estabiliza a taxa de erro antes dos primeiros 130 quadros, o que representa um perído de $4,33$ segundos para uma taxa de 30 quadros por segundo e $2,16$ segundos para uma taxa de 60 quadros por segundo, que é a tendência para vídeos de alta resolução em telas de grandes dimensões. Os resultados mostraram que o algoritmo pro-

posto reduz a complexidade computacional, ou seja, o tempo de processamento do processo em 30% em média em relação ao codificador de referência. Além as perdas de qualidade objetiva e a eficiência de compressão medidas pelas métricas de Bjøntegaard apresentaram valores desprezíveis em relação aos resultados da implementação de referência e está alinhada com os resultados mais atuais da literatura revisada.

No Capítulo 2 é feita uma revisão bibliográfica da literatura referente a propostas de otimização de processos de codificação de vídeo digital. Essa revisão mostrou que a abordagem está bem alinhada com o andamento das pesquisas recentes da comunidade científica especializada nesse tema. No Capítulo 3 é descrito o processo de padronização e, detalhadamente, o funcionamento do codificador HEVC. O Capítulo 4 apresenta o conceito de aprendizagem de máquina e explica as técnicas usadas com enfoque para a máquina de vetores suporte e uma de suas principais otimizações, o algoritmo Pegasos, que é configurado para execução em tempo de execução. No Capítulo 5 é apresentado o algoritmo proposto, que será chamado de Codificador $\Omega$ no andamento desse trabalho. O Captulo 6 descreve a metodologia utilizada, que é fortemente baseada no documento (BOSSEN, February, 2012), e as ferramentas aplicadas nas análises e implementações. Após isso, as simulações e análises estão descritas no Capítulo 7 e, finalmente, os resultados, conclusões e propostas de trabalhos futuros são apresentados no Capítulo 8.

# CAPÍTULO 2

# Revisão Bibliográfica

A maioria dos trabalhos revisados sobre o HEVC utiliza o codificador de referência do ITU (*International Telecommunication Union*), chamado HM (HEVC Test Model), como base para aplicar as mudanças necessárias analisada por cada estudo. Poucos estudos (HU *et al.*, 2014) utilizam o codificador x265. O x265 é codificador de código aberto desenvolvido e mantido pelo FFmpeg (FFMPEG..., ) e apesar de apresentar um desempenho melhor, em termos do tempo de execução, implementa apenas um conjunto restrito das especificações do HEVC, o que inviabilizaria a análise comparativa da influência dos módulos definidos na especificação no desempenho geral do codificador (ZUPANCIC *et al.*, 2015a).

Uma solução para a redução da quantidade de esforço computacional é usar os dados da sequência de vídeo em tempo de codificação. Abordagens de processamento das informações do sinal de vídeo geradas em tempo de codificação são recentes e menos exploradas em comparação com o processamento no domínio espacial antes ou depois do processo da codificação (KIM *et al.*, 2015). Um exemplo de um algoritmo simples e rápido para a detecção de alterações em tempo de codificação é apresentado em (BRACAMONTE *et al.*, 2005) e (SZCZERBA *et al.*, 2009).

A aprendizagem de máquina é um conjunto de técnicas e algoritmos aplicados em inteligência artificial, reconhecimento de padrões e processamento de sinal, uma vez que aprende a partir de grandes volumes de dados em situações complexas e tem a capacidade de poder chegar à solução ideal em algumas situações.. Com esta propriedade, os pesquisadores tentam aplicar algoritmos de aprendizagem automática na codificação de vídeo para atingir uma melhor eficiência de compressão. O artigo (LAMPERT, 2006), publicado em 2006, é um dos primeiros trabalhos que descreve de forma mais clara o uso de técnicas de aprendizado de máquina para otimizar o processo de codificação de vídeo. Nesse trabalho ainda é ainda abordado o codificador H.264. Em (VAN *et al.*, 2013), é feito uso de técnicas de classificação, mais especificadamente o algoritmo de classificação estatística por árvore de decisão, chamado C4.5, para aplicações de transcodificação de taxa para codificadores HEVC.

Métodos de aprendizagem automática em tempo real (*Online Learning*) são bem correlacionados com métodos de gradiente estocástico por operaram em apenas uma amostra do conjunto de treinamento, em cada iteração. Essa abordagem é altamente recomendada quando o problema aborda grandes volumes de dados. Essa argumentação foi fundamental para o problema abordado nesta tese, dado que a quantidade de amostras treinadas ultrapassa a ordem dos milhões em menos de três segundos. O armazenamento de tais valores para treinamento seria inviável, por exemplo, para dispositivos embarcados, nos quais a capacidade de armazenamento é baixa. O algoritmo Pegasos (*Primal Estimated sub-GrAdient SOlver for SVM* (SHALEV-SHWARTZ *et al.*, 2007) compartilha a simplicidade e a velocidade que são características de algoritmos de aprendizagem automática em tempo real e ainda garante a convergência para a solução ótima, baseando-se no algoritmo SVM (*Support Vector Machine*). O algoritmo Pegasos é uma aplicação de um método de gradiente estocástico. No contexto de problemas de aprendizagem automática, a eficiência da abordagem do uso de gradiente estocástico foi observada experimentalmente que, algoritmos estocásticos produzem o melhor desempenho de generalização (SHALEV-SHWARTZ *et al.*, 2007). Os trabalho publicado por (TOHIDYPOUR *et al.*, 2015) mostra uma aplicação de técnica de aprendizagem automática em tempo real para codificação HEVC de vídeos em três dimensões onde é explorado a correlação entre características dos mapas de profundidade e do vídeo base. Foram utilizadas informações de textura e dos vetores de movimento. A abordagem resultou em reduções de tempo de processamento de 67% para os mapas de profundidade e 27% ao todo.

Considerando o problema de compressão de vídeo de elevadas resoluções, nomeados de vídeos em UHD (*Ultra High Definition*), em (NACCARI *et al.*, 2015) é proposta uma análise do desempenho do codificador HEVC focado em aplicações para UHDTV (*Ultra High Definition Television*), avaliando cada ferramenta de codificação e o impacto delas na eficiência de compressão e complexidade computacional. Dá-se o nome de ferramenta de codificação aos diferentes algoritmos e processos que combinados produzem um fluxo de bits codificados a partir do vídeo original. Dessa forma, é proposta uma configuração compatível com o cenário atual da capacidade de transmissão. Além disso três novas técnicas de otimização são introduzidas, que resultaram em $11,5\%$ de redução do tempo de processamento com um aumento de $3,1\%$ na taxa de *bits*. Outro trabalho que aborda a codificação de vídeos UHD foi realizado em (PODDER *et al.*, 2015). Nesse artigo foi feita uma análise do processo de codificação HEVC para vídeos UHD e mostrou-se que a etapa de predição temporal e a estimação de movimento são responsáveis por uma percentagem considerável do tempo de codificação especialmente quando o objetivo é a compressão de conteúdo UHD. Um novo algoritmo para reduzir a complexidade computacional foi proposto com o objetivo de endereçar o término prematuro do particionamento das unidades de codificação. Dentre os trabalhos analisados, essa abordagem apresentou um dos piores resultados reduzindo apenas $9,8\%$ do tempo de codificação quando comparados com a execução de um codificador de referência.

Em outro abordagem, que foca em vídeos UHD, o artigo (ZUPANCIC *et al.*, 2015a) estuda o desempenho da implementação de referência do codificador HEVC para vídeos UHD, analisando, em termos de tempo gasto, a realização das diferentes etapas da codificação. Com base nesta análise, um novo algoritmo é apresentado para reduzir a complexidade da etapa de predição Inter, restringindo os modos a serem testados e usando técnicas de aceleração da etapa de estimação de movimento. Os resultados experimentais mostram que o algoritmo proposto alcança, em média, uma economia de tempo de $72,9\%$, para perdas de codificação limitadas. Em (LI *et al.*, 2015) um novo algoritmo,também focado na etapa de estimação de movimento, é proposto tendo capacidade adaptativa ao contexto usando algoritmos de medição da intensidade de movimento, baseando-se nas informações dos magnitude dos vetores de movimento e nas diferenças entre vetores de blocos vizinhos. Dessa forma, quando a intensidade de movimento é elevada, o algoritmo TZ *Search* é utilizado, de outro modo, quando os valores são reduzidos, o algoritmo da busca hexagonal é aplicado. Os resultados experimentais publicados mostraram que o método proposto reduz de 30% a 60% o tempo do processo de estimação de movimento, com reduções na eficiência de codificação, em média, de $0,5\%$ BD-Rate. A métrica BD-Rate faz parte de conjunto de métricas desenvolvidas por Gisle Bjøtegaard e serão explicas no Capítulo 4.

Em (ZHANG *et al.*, 2015a), diferente de outros métodos que se concentram em obter os modos mais prováveis de acordo com o limiar de custo ou com os modos de blocos vizinhos no espaço e no tempo, esse trabalho propõe determinar os modos menos prováveis baseando-se na distribuição da distorção de forma a acelerar o processo de codificação com pequenas perdas de qualidade. Foi obtida uma correlação entre a distorção e o modo de predição Inter. Essa abordagem alcançou um expressivo resultado, reduzindo a complexidade computacional em até 77% em comparação ao codificador de referência HM 16.4. Considerando a ativação de ferramentas nativas de decisão prematura de particionamentos desse codificador, as reduções também foram significativas, resultando em 48% de redução do tempo de codificação com um aumento de $2,9\%$ na taxa de *bits*.

Existem alguns trabalhos que abordam algoritmos para a determinação prematura do particionamento das unidades de codificação. Geralmente, os algoritmos propostos fazem uso de informação previamente adquirida. Em (XIONG *et al.*, 2014b) é um proposto um método baseado em uma técnica, conhecida como divergência de movimento piramidal (PMD – *Pyramid Motion Divergence*), para avaliar o modo Skip para cada formato de unidade de codificação (*Coding Unit* – CU) inter-quadros avaliado. O método PMD avalia a variância do fluxo óptico da unidade de codificação corrente em relação aos sub-CUs. Também é utilizado o algoritmo *k Nearest Neighboring* (*k*NN) para determinar o modo de divisão de cada unidade de codificação. Nesse trabalho é feito um cálculo relativamente simples, mas de qualquer forma, há impacto na redução da complexidade computacional. Em (YU *et al.*, 2013), a etapa de decisão do modo Skip de um codificador H.264 para vídeos estéreos é modelada como um problema de classificação, usando uma árvore de decisão. Foi observado que para esse tipo de vídeo, o modo Skip

é significativamente mais frequente que outros modos. No entanto, esses algoritmos que abordam o H.264 não adequados para serem aplicados em problemas análogos no HEVC devido a diferença entre as características estatísticas desses codificadores.

O trabalho (DENG *et al.*, 2014) aplica um mapeamento de ROI (*Region of Interest*) para definir antecipadamente o nível de profundidade da avaliação RD de um CTU de forma a controlar a complexidade computacional do processo de codificação. O modelo de atenção ROI é baseado nas características do sistema visual humano. De acordo com o modelamento dessas características, quando uma pessoa assiste a um vídeo genérico, ela não deve prender sua atenção em toda a cena, mas somente a uma pequena região em volta de um ponto de fixação, conhecida como região de interesse. Um algoritmo de classificação é descrito em (MALLIKARACHCHI *et al.*, 2014) baseado no conceito de homogeneidade de movimento de uma unidade de codificação e na taxa de distorção, para encontrar o melhor modo de predição inter para encerrar a busca prematuramente. Esse algoritmo mostrou bons resultados, chegando a significativos 73,25% de redução de complexidade computacional, um dos melhores resultados encontrados nas pesquisas para essa TS. Uma nova característica baseada no cálculo da PVAD (*Pyramid Variance of Absolute Difference*) é utilizada como parâmetro de um método conhecido como *Graph Cut* em (XIONG *et al.*, 2014a).

Em uma abordagem semelhante à proposta nesta tese, o artigo (AHN *et al.*, 2015) é apresentado um esquema de decisão prematura do modo Skip e um método de particionamento rápido da unidade de codificação. Esses esquemas apresentaram perdas negligenciáveis em relação a efiência de compressão. A abordagem utilizada consiste em utilizar parâmetros que mapeiam as características espaço-temporais da sequência de vídeo que estão disponíveis durante o processo de codificação HEVC sem a necessidade de produção de nenhum atributo adicional que possa demandar um custo computacional adicional. Essa análise utilizou os parâmetros da etapa de filtragem SAO (*Sample Adaptative Offset*), tamanho da TU (*Transform Unit*), vetores de movimento e a informação da quantidade de coeficientes zerados antes da quantização. Os resultados mostraram uma redução de 42%, em média, no tempo total de processamento com perdas de, no máximo, $1,4\%$. O trabalho realizado em (SHEN *et al.*, 2014) também explorada características espaço-temporais, tendo como principal idéia a exploração da correlação entre o nível de profundidade do processo de particionamento e informações espaço-temporais do conteúdo para ajustar a decisão do modo inter. Em (LEE *et al.*, 2015a) é também abordado o mesmo problema da tese, e embora não tenha alcançado uma redução de complexidade computacional tão expressiva quanto em outros trabalhos, os resultados mostraram que a eficiência de compressão foi aprimorada na maioria dos vídeos testados, principalmente para configuração *Low Delay B*. A análise desse artigo mostrou que existe uma forte correlação entre o particionamento do bloco e as medidas de distorção calculadas no processo.

Na linha de codificadores perceptuais de vídeo (PVC – *Perceptual Video Coding*), o trabalho realizado em (KIM *et al.*, 2015) é proposto um esquema de codificação baseado em modelos que usam a métrica JND (*Just Noticeable Distortion*). Esse modelo parametriza a etapa

de transformação considerando características como a sensibilidade de contraste, adaptação de luminância e efeitos de mascaramento por contrastes. Os resultados mostraram que essa abordagem obteve bons resultados na redução da taxa de *bits* resultante de, em média 16%, para a configuração *Random Access*

O trabalho (CORREA *et al.*, 2015) apresenta os primeiros resultados do estudo apresentado em (CORREA *et al.*, 2014). Nesse trabalho foi desenvolvido um método de término prematuro da definição da estrutura de cada CTU da sequência de vídeo. Esse método aplica árvores de classificação obtidas usando técnicas de mineração de dados. As árvores de classificação foram treinadas usando resultados intermediários do processo de codificação de um conjunto de sequências de vídeo de testes e, depois, implementadas em um codificador de referência HM com o propósito de abreviar todo o processo de otimização distorção-taxa. Devido ao grande volume de dados gerado no processo de codificação de sequências em ultra alta definição, algoritmos de mineração de dados têm ajudado a pesquisar quais variáveis do processo podem ajudar na tomada de decisão dentro do próprio codificador. De maneira geral, essas técnicas apresentam-se como tendência para gerações futuras do codificador.

A pesquisa realizada por (ZHANG *et al.*, 2015b) executa uma classificação conjunta que consiste na utilização de mútiplos classificadores SVM (*Support Vector Machine)*. Desse trabalho, foi retirada a fundamentação para o uso de algoritmos baseados em SVM nesta tese. Com essa estrutura foi possível garantir um processo flexível de decisão da profundidade da unidade de codificação, acarretando ganhos em relação ao processo de RDO convencional com uma boa relação custo-benefício entre a perda de qualidade e a complexidade computacional do algoritmo. Em (PAN *et al.*, 2014) é aplicado um algoritmo para decisão antecipada da aplicação do modo de fusão de blocos (MERGE *mode*) usando informações como a correlação entre os modos de uma CU raiz e suas CU filhas, AZB (*All Zero Block* e variáveis da estimação de movimento. Esse algoritmo demonstrou uma redução média de 58,96% no tempo de processamento com pequenas perdas de 0,32% BDBR. Outro trabalho que utiliza SVM foi proposto por (ZHANG *et al.*, 2015b) que executa uma classificação conjunta, que consiste na utilização de mútiplos classificadores SVM (*Support Vector Machine)*. Com essa estrutura foi possível garantir um processo flexível de decisão da profundidade da unidade de codificação, acarretando em ganhos em relação ao processo de RDO convencional com uma boa relação custo-benefício entre a perda de qualidade e a complexidade computacional do algoritmo.

Em (MALLIKARACHCHI *et al.*, 2014) é desenvolvido um algoritmo de classificação baseado na homogeneidade de movimento de uma unidade de codificação e na taxa de distorção para encontrar o melhor modo de predição inter encerrando a busca prematuramente. Esse algoritmo mostrou bons resultados, chegando a significativos 73,25% de redução de complexidade computacional. Um dos poucos estudos que aborda todas as etapas do processo de otimização da relação taxa/distorção (RDO) do codificador HEVC é proposto em (CORREA *et al.*, 2015). Nesse estudo foi desenvolvida uma ferramenta que usa a técnica de mineração de dados para construir um conjunto de árvores de decisão que possibilitam o fim prematuro do processo deci-

sório para encontrar a melhor configuração para unidades de codificação, unidades de predição e unidades de transformação. Essa abordagem apresenta uma redução da complexidade computacional de até 88% em relação ao codificador de referência HM, reforçando a necessidade de uma combinação de abordagens de otimizações para obtenção dos melhores resultados.

No Artigo (AHN *et al.*, 2015), é apresentado um esquema de codificação da unidade de codificação com perdas de eficiência de codificação desprezíveis. Ele consiste em um algoritmo de detecção do modo Skip. Para isso, são usados os parâmetros de codificação espaço-temporal que já estão disponíveis durante o processo de codificação HEVC, sem a necessidade de nenhum processamento adicional para a criação de novos atributos. Os parâmetros utilizados nesse algoritmo foram: Parâmetros da etapa SAO (*Sample Adaptative Offset*), tamanhos das unidades de transformção (TU – *Transformation Unit*), tamanhos dos vetores de movimento e a informação CBF (*Coded Block Flag*), que considera que todos os pixels residuais desse bloco possuem valor 0.

Outros pontos podem ser explorados na tentativa de redução de complexidade computacional e aumento da eficiência de compressão. A etapa de estimação de movimento é um dos principais pontos em que ganhos podem ser obtidos, principalmente porque, em geral, os algoritmos de busca, inerentemente, pode apresentar um problema de erro mínimo local, quando seu resultado converge para uma solução não-ótima dentro da janela de busca definida. De maneira geral, não há garantias de que a posição obtida nas buscas seja ótima. Em (PAN *et al.*, 2013) são propostas duas estratégias de término prematuro do processo de estimação de movimento do algoritmo TZSearch implementado no codificador de referência HM. O processo baseia-se em uma análise estatística da probabilidade de selecionar o vetor preditor mediano como melhor busca para os diferentes tamanhos da unidade de codificação. Em (WANG *et al.*, 2014), os autores utilizam a correlação entre o nivel de profundidade da arvóre e o número de coeficientes diferentes de zero nesse nível para determinar o melhor tamanho da unidade de codificação e finalizar o processo de avaliação RD. No entanto, essa abordagem não se mostrou eficiente para sequências com movimentos rápidos e com texturas mais elaboradas. O modo de predição Intra também apresenta oportunidades de otimização em seu processo.

Em (ZENG *et al.*, 2011) é definida uma metodologia de análise chamada *Correlation-Based Mode Decision* (CBMD) para acelerar o processo de codificação, diminuindo o número de modos Intra testados durante a etapa de otimização taxa-distorção (*Rate Distortion Optimization*). Os modos de predição intra são categorizados em cinco classes de movimentação e atividades diferentes, e uma delas é definida como o conjunto ótimo de modos para cada caso. Em (FANG *et al.*, 2013) é desenvolvido um classificador bayesiano que utiliza um conjunto de características previamente escolhidas atráves de uma análise discriminante. Dessa forma, é possível variar o tamanho desse conjunto usado como entrada do classificador, dimininuido o tempo de processamento necessário. Em (LEE *et al.*, 2015b) é usada uma técnica conhecida como *Local Binary Pattern* que extrai a informação da textura local para determinar o modo de predição Intra do PU corrente.

O artigo (JIA *et al.*, 2006) revisa as métricas de avaliação de qualidade objetiva mais comuns e apresenta a métrica JND (*Just-Noticeable Distortion*). Em (SESHADRINATHAN *et al.*, 2010) são apresentados os resultados de um estudo de grande escala de avaliação subjetiva de qualidade de vídeos aplicando diversos tipos de distorções nos vídeos estudados. Esses resultados foram utilizados para testar o desempenho dos algoritmos de avaliação de qualidade objetiva mais utilizados. Os resultados desses estudos e os dados coletados estão disponíveis numa base de dados batizada de *Laboratory for Image and Video Engineering* (LIVE) *Video Quality Database*. Em (YU; WINKLER, 2013) são aprofundados os conceitos de complexidade espacial e temporal que baseiam o processo de codificação de vídeo. O trabalho (KORHONEN *et al.*, 2013) analiza a relação entre o resultado de avaliação subjetiva com algumas características do contéudo exibido, estudando o impacto das distorções espaciais e temporais na qualidade total percebida pelo usuário.

Embora o artigo (TAN *et al.*, 2015) relate um extenso conjunto de resultados de testes de verificação da codificação HEVC, também resume detalhes de ferramentas que podem ser utilizadas na análise dos resultados que apontam para fatores que devem se considerados uma avaliação, como por exemplo a métrica Bjøntegaard. Os resultados apresentados neste trabalho são baseados no uso de mais espectadores para testes subjetivos, resultados objetivos são apresentados e comparados com os resultados subjetivos e análises adicionais dos ganhos de codificação em relação a taxa de *bits* são fornecidos. Em última análise, o teste de verificação mostrou que o objectivo principal do HEVC é alcançado, isto é, proporcionando uma melhoria substancial na eficiência de compressão em relação ao seu antecessor AVC.

Para desenvolver a base de conhecimento sobre codificação de vídeo HEVC foi realizado um detalhado estudo sobre a sua arquitetura, como apresentado no próximo capítulo.

# CAPÍTULO 3

# Tecnologias e Padrões de Vídeo Digital

A etapa de codificação transforma o vídeo de entrada em uma fluxo de *bits* que é um representação codificada do vídeo original. O fluxo de *bits* dos padrões de vídeo atuais usa uma estrutura de pacotes em que a informação codificada é organizada hierarquicamente de acordo com a especificação abordada. O codificador deve gerar um fluxo de *bits* em conformidade com as necessidades e requisitos da aplicação. Isso inclui a seleção de ferramentas (que são determinadas pelo perfil de codificação escolhido) ou restrições na disponibilidade de memória no processo de decodificação e a taxa de transmissão resultante do vídeo codificado.

A otimização do controle das decisões de codificação é um fator importante do projeto do codificador. Dependendo do tempo disponível para codificar o vídeo, a quantidade e os esforços de otimização variam significativamente. Para aplicações de tempo real, por exemplo, o codificador usa apenas informações passadas do vídeo corrente para aprimorar a predição. Além disso, os requisitos da transmissão e outros eventos imprevisíveis influenciam no processo de codificação. Em sistemas de produção de conteúdo, que codificam o vídeo para armazenamento ou distribuição por mídias, dispendiosos processamentos são executados devido à não haver necessidade de transmissão em tempo real. Sendo assim diversos processos de otimização são habilitados, assim como a otimização da parametrização.

## 3.1 O Padrão ITU-T H.264

O principal objetivo do projeto H.264 foi desenvolver um padrão de codificação de vídeo de alto desempenho adotando técnicas fundamentais de projeto simples e que tivessem seu desempenho previamente comprovado. O grupo ITU-T VCEG (*Video Coding Experts Group*) iniciou o trabalho de padronização em 1997. O projeto do H.264 oferecia uma série de vantagens quando comparados com os padrões de codificação disponíveis na época. Podem-se destacar as seguintes (ZENG ABDUL REHMAN *et al.*, 2013):

- Mais de 50% de economia de taxa de transmissão de *bits*: Comparando-o ao MPEG-2 e ao MPEG-4 *Simple Profile*, o H.264 consegue uma redução de 50% em taxa de transmissão considerando o nível de parametrização similar.
- Vídeo em alta qualidade: Alta qualidade de vídeos para baixas e altas taxas de codificação.
- Resiliência a erros: O H.264 implementa técnicas avançada para tratar erros por perdas de pacotes e erros de *bit*.
- Amigável a redes: A camada NAL (*Network Adaptation Layer)* abstrai a camada física do processo de codificação e oferece suporte a diferentes tipos de rede de transmissão.

O processo de codificação do H.264 utiliza a abordagem de blocos, também utilizadas nas gerações anteriores. O processo básico é composto das seguintes etapas:

1. Divisão de cada quadro do vídeo em blocos de *pixels*, que serão as entidades básicas de processamento no processo de codificação.
2. Exploração da redundância espacial dentro de um quadro de vídeo usando as etapas de predição espacial, transformação de domínio, quantização e codificação de entropia.
3. Exploração da redundância temporal existentes em blocos de mesma posição em quadros sucessivos, para que, dessa forma, apenas as diferenças entre eles precisem ser codificadas. Esse processo é realizado pelas etapas de estimação e compensação de movimento. O processo de estimação de movimento é detalhadamente descrito na sequência desse texto.
4. Exploração da redundância espacial restante nos quadros resultantes da estimação de movimento.

Além do modo de codificação progressivo onde os blocos são processados pela sequência de linhas do quadro, o processo de codificação H.264 suporta o modo entrelaçado. Nesse modo um quadro é dividido em dois campos. Campos podem ser codificados usando entrelaçamento espacial ou temporal (ZENG ABDUL REHMAN *et al.*, 2013).

### 3.1.1 Tipos de *Slices*

O padrão H.264 define cinco diferentes tipos de *slices*: I, P, B, SI e SP. *Slices* I, ou *Slices* Intra-quadros, descrevem uma imagem completa, contendo referências a ela mesma. Um vídeo H.264 pode conter apenas quadros I, mas isso é raramente utilizado. O primeiro quadro de uma sequência é obrigatoriamente sempre I. *Slices* P, ou *slices* preditos, usam um ou mais *slices* recentemente decodificados como referência para reconstrução do quadro. A predição geralmente não é exatamente igual ao quadro atual, dessa forma, um erro residual é adicionado

ao processo. *Slices* B, ou *Bi-predicted slices*, funcionam assim como os *slices* P, sendo que usam uma ou mais referências futuras, além das referências passadas. Dessa forma, *slices* B precisam ser codificados antes de um *slice* I e um *slice* P. *Slices* SI e SP são usados para realizar transições entre dois fluxos H.264 diferentes.

### 3.1.2 Perfis e Níveis

O padrão ITU-T H.264 define um conjunto de três perfis. Um perfil de codificação define os requisitos do codificador ou decodifador e especifica um conjunto de ferramentas de codificação de acordo com a finalidade do perfil. São eles (WIEGAND *et al.*, 2003):

- *Baseline profile* – oferece suporte à codificação intra-quadros e inter-quadros (usando *slices* I e P respectivamente) e codificação de entropia com códigos de comprimento variável de contexto adaptativo (*Context-Adaptive Variable-Length Codes* – CAVLC). Aplicado em videotelefonia, videoconferência e comunicações sem fio.

- *Main profile* - suporta vídeo entrelaçado, codificação inter-quadros usando *slices* do tipo B, codificação inter-quadros usando perdição ponderada e codificação aritmétrica de contexto adaptativo (*Context-Based Arithmetic Coding* – CABAC). Aplicado em radiodifusão televisiva e armazenamento de vídeo.

- *Extended profile* – não suporta vídeo entrelaçado e CABAC, mas oferece suporte a métodos de permutação de diferentes fluxos de *bits* codificados e resistência e recuperação de erros aprimorada. Usado principalmente em aplicações de *streaming* de mídias.

Os limites de desempenho de codificadores e decodificadores são definidos por um conjunto de níveis. Atualmente o padrão H.264 define um conjunto de 16 níveis de codificação, como mostrado na Tabela 3.1 (STOCKHAMMER *et al.*, 2003). Os itens da tabela com os nomes dos perfis correspondem à máxima taxa de transmissão de *bits* alcançada pelo perfil especificado em determinado nível (WIEGAND *et al.*, 2003):

- **MB/s max** – número máximo de macroblocos por segundo;

- **FS max** – dimensão máxima do quadro;

- **res@FR** – exemplo de resolução a determinada taxa de quadros por segundo.

O nível também define o tamanho da memória de armazenamento de entrada do decodificador.

### 3.1.3 Extensões ao Padrão Original

Mesmo tendo uma vasta cobertura de tipos de aplicações possíveis, o padrão H.264/AVC é basicamente focado em qualidade para vídeo de entretenimento, baseado em 8 *bits* por amostra e formato de subamostragem de crominância 4:2:0. Esse formato define a amostragem da

**Tabela 3.1** Tabela de níveis do padrão H.264.

| **Nível** | **MB/s max** | **FS max** | ***Extended, Main, Baseline*** | ***High Profile*** | ***High 10 Profile*** | ***High Profile 4:2:2 e 4:4:4*** | **res@FR** |
|---|---|---|---|---|---|---|---|
| 1 | 1485 | 99 | 64 kbit/s | 80 kbit/s | 192 kbit/s | 256 kbit/s | 128x96@30.9 |
| 1b | 1485 | 99 | 128 kbit/s | 160 kbit/s | 384 kbit/s | 512 kbit/s | 128x96@30.9 |
| 1.1 | 3000 | 396 | 192 kbit/s | 240 kbit/s | 576 kbit/s | 768 kbit/s | 320x240@10.0 |
| 1.2 | 6000 | 396 | 384 kbit/s | 480 kbit/s | 1152 kbit/s | 1536 kbit/s | 320x240@20.0 |
| 1.3 | 11880 | 396 | 768 kbit/s | 960 kbit/s | 2304 kbit/s | 3072 kbit/s | 352x288@30.0 |
| 2 | 11880 | 396 | 2 Mbit/s | 2.5 Mbit/s | 6 Mbit/s | 8 Mbit/s | 352x288@30.0 |
| 2.1 | 19800 | 792 | 4 Mbit/s | 5 Mbit/s | 12 Mbit/s | 16 Mbit/s | 352x576@25.0 |
| 2.2 | 20250 | 1620 | 4 Mbit/s | 5 Mbit/s | 12 Mbit/s | 16 Mbit/s | 720x480@15.0 |
| 3 | 40500 | 1620 | 10 Mbit/s | 12.5 Mbit/s | 30 Mbit/s | 40 Mbit/s | 720x480@30.0 |
| 3.1 | 108000 | 3600 | 14 Mbit/s | 17.5 Mbit/s | 42 Mbit/s | 56 Mbit/s | 1280x720@30.0 |
| 3.2 | 216000 | 5120 | 20 Mbit/s | 25 Mbit/s | 60 Mbit/s | 80 Mbit/s | 1280x1024@42.2 |
| 4 | 245760 | 8192 | 20 Mbit/s | 25 Mbit/s | 60 Mbit/s | 80 Mbit/s | 2048x1024@30.0 |
| 4.1 | 245760 | 8192 | 50 Mbit/s | 62.5 Mbit/s | 150 Mbit/s | 200 Mbit/s | 2048x1024@30.0 |
| 4.2 | 522240 | 8704 | 50 Mbit/s | 62.5 Mbit/s | 150 Mbit/s | 200 Mbit/s | 2048x1088@60.0 |
| 5 | 589824 | 22080 | 135 Mbit/s | 168.75 Mbit/s | 405 Mbit/s | 540 Mbit/s | 3680x1536/26.7 |
| 5.1 | 983040 | 36864 | 240 Mbit/s | 300 Mbit/s | 720 Mbit/s | 960 Mbit/s | 4096x2304@26.7 |

componente de luminância e a amostragem vertical e horizontal para das crominâncias verticais e horizontais. Devido a suas limitições de tempo de desenvolvimento, o padrão inicial não incluiu suporte para a maioria dos ambientes profissionais de edição e produção de vídeo e também não teve seu desenvolvimento voltado às mais elevadas resoluções de vídeo. Para resolver esses problemas, uma continuação dos trabalhos do projeto em conjunto foi iniciada para adicionar novas extensões às capacidades do padrão original (ALENCAR, 2007).

Esse novo esforço teve ínicio em maio de 2003, tendo seus trabalhos finalizados em julho de 2004. A divulgação oficial das novas funcionalidades do padrão H.264/AVC FRExt aconteceu em setembro de 2004. Essa extensão originalmente conhecida como Extensão Profissional foi renomeada como *Fidelity Range Extensions* para melhor refletir o intuito das ferramentas adicionadas. Com o tempo adicional foi possível incluir:

- Suporte ao dimensionamento adaptativo do tamanho dos blocos para a transformada espacial;
- Suporte a um modelo de adaptação perceptual das matrizes de quantização escaláveis;
- Um conjunto de quatro novos perfis conhecidos como *High Profiles*:
  - High profile (HP) – possui suporte à representação de 8 *bits*/símbolo com subamostragem de crominância 4:2:0, visa abranger aplicações que demandem alta definição sem necessidade de um formato de croma maior ou melhor precisão nas amostras de vídeo.
  - High 10 Profile (Hi10P) – 4:2:0 e até 10 *bits*/amostra.
  - High 4:2:2 Profile (H422P) – 4:2:2 e até 10 *bits*/amostra.

– High 4:4:4 Profile (H444P) – 4:4:4 e até 12 *bits*/amostra. Otimizado para codificação no espaço de cores RGB de forma a evitar erros de conversão.

### 3.1.4 Codificação Intra-quadros

A codificação Intra-quadros se refere às etapas nas quais apenas a redundância espacial da informação de vídeo é explorada e se aplica apenas aos quadros e *slices* I. Os quadros I são tipicamente codificados aplicando a operação de transformada aos blocos do quadro. Devido a explorar a apenas a redundância espacial, a representação codificada de quadros I possui maior quantidade de dados do que os outros tipo de quadros que executam as etapas de exploração temporal.

O princípio da codificação baseia-se na observação de que blocos vizinhos tendem a ter propriedades semelhantes. Dessa forma, o bloco alvo é previsto a partir de algum dos seus vizinhos. A diferença entre o bloco atual e sua predição é calculada e submetida as próximas etapas de codificação.

### 3.1.5 Codificação Inter-quadros

A codificação e predição Inter-quadros consiste em utilizar as técnicas de compensação e estimação de movimento para exploração da redundância temporal entre quadros sucessivos de um vídeo. A estimação de movimento no H.264 suporta a maioria das características dos padrões anteriores, adicionando uma maior flexibilidade e novas funcionalidades aos algoritmos suportados. O padrão H.264 permite que os vetores de movimento sejam determinados com maiores níves de precisão espacial, aprimorando capacidade de predição dos algoritmos de estimação de movimento. A maior precisão é feita com os processos de estimação de movimento com precisão de $1/4$ de *pixel*.

## 3.2 O Padrão HEVC

### 3.2.1 Introdução

O padrão HEVC (*High Efficiency Video Coding*) é projetado para alcançar várias metas, incluido a alta eficiência de codificação, facilidade de integração de sistemas de transporte e resiliência em perda de informações, assim como a adaptabilidade para arquitetura de *hardware* com processamento paralelo. São descritos nas próximas seções os principais pontos da arquitetura do padrão HEVC, com destaque para as novas funcionalidades, e as operações típicas para a produção do fluxo de *bits* válido.

### 3.2.2 O Processo de Padronização do HEVC

Depois da finalização da especificação do H.264/AVC High Profile em meados de 2004, os comitês ITU-T VCEG e ISO/IEC MPEG começaram a tentar identificar quando os próximos principais avanços em compressão de vídeo estariam prontos para padronização (OHM; SULLIVAN, 2013). Em 2004, o comitê VCEG começou a estudar tecnologias potenciais, em 2005 iniciou as investigações de áreas de tecnologias chave e, após essas investigações, prôpos uma base de código comum derivada do código do codificador de referência do H.264/AVC. Entre 2005 e 2008, deu-se início a atividades exploratórias em busca de aprimoramentos significativos no processo de compressão e vários *workshops* foram realizados para discutir esses aprimoramentos. Em 2008 uma chamada CfE (*Call for Evidences*) foi publicada para receber novas propostas de aprimoramentos. Testes subjetivos foram realizados para avaliar os resultados das submissões.

A partir das investigações dos trabalhos submetidos, os dois comitês concordaram que já haviam tecnologias suficientes com potencial de aprimoramentos significativos em eficiência de compressão quando comparadas a versão corrente do H.264/AVC. Em Janeiro de 2010, os dois comitês estabeleceram o grupo JCT-VC (*Joint Collaborative Team on Video Coding*) e, no mesmo ano, uma chamada de propostas (CfP – *Call for Proposals*) para tecnologias de compressão de vídeo fora lançada para identificar as tecnologias iniciais que serviriam de base para as futuras atividades de padronização. No primeiro encontro do grupo, em abril de 2010, as propostas submetidas forem estudadas e uma primeira versão, chamada de TMuC (*Test Model under Consideration*), foi definida a partir de elementos das propostas mais promissoras. Além disso, o nome *High Efficiency Video Coding* foi escolhido. As tecnologias submetidas nas principais contribuições propostas foram anteriormente discutidas em uma edição especial do periódico *IEEE Transactions on Circuits and Systems for Video Technology* (OHM; SULLIVAN, 2013).

Mesmo que a TMuC tenha mostrado avanços significativos em eficiência de compressão, ainda havia um número substancial de funcionalidades de codificação redundantes, principalmente devido ao fato de o código de referência ser formado a partir de várias contribuições que foram apressadamente agrupadas. A versão 1 do modelo de teste do HEVC (HM 1) e a versão 1 do rascunho de especificação correspondente foram produzidos como resultados do terceiro encontro do grupo JCT-VC em outubro de 2010. Comparado ao projeto dos TMuC anteriores, o HM 1 foi significativamente simplificado em função da remoção de ferramentas de codificação que demonstração ganhos de compressão mínimos em relação a suas respectivas complexidades computacionais e complexidade computacional do sistema (WIEGAND *et al.*, 2012). Vários outros aspectos, além de aprimoramentos em compressão, foram investigados entre o quarto e o décimo-primeiro encontro do JCT-VC, incluindo redução de complexidade computacional, unificação de várias ferramentas de codificação e a identificação de elementos passíveis de processamento paralelo (TOHIDYPOUR *et al.*, 2015). O desenvolvimento do padrão proposto chegou

a seu primeiro marco significativo em Fevereiro de 2012 com o término do primeiro rascunho parcial da especificação HEVC (NIGHTINGALE *et al.*, 2012). O projeto do HEVC foi continuamente aprimorado até o lançamento da versão final em Janeiro de 2013. Para o ISO/IEC, o HEVC é chamado *International Standard* 23008-2 ou MPEG-H Part 2, e, para o ITU-T, deve ser padronizado como *Recommendation* H.265. A principal meta do JCT-VC era projetar e desenvolver a próxima geração de padrões de codificação de vídeo que oferecesse um aprimoramento na taxa de compressão de 50% em relação ao padrão H.264/AVC sem perdas na qualidade visual percebida (OHM *et al.*, 2012).

O principal padrão de codificação de vídeo que precedeu o projeto do HEVC foi o padrão H.264/AVC, desenvolvido no intervalo entre 1999 e 2003, que, posteriormente, fora extendido de várias outras formas de 2003 a 2009. O padrão H.264/AVC é uma tecnologia que possibilita aplicações de vídeo digital em áreas que não eram antes cobertas pelos seus antecessores, entre eles, o MPEG-2. Ele tem sido usado em transmissões de televisão em alta definição por satélite, cabo e difusão terrestre, sistemas de edição e aquisição de vídeo, filmadoras, aplicações de segurança, Internet, vídeo para dispositivos móveis, discos Blu-ray e aplicações de teleconferência ao vivo. No entanto, uma crescente diversidade de serviços, a popularidade crescente do vídeo em altas definições e o surgimento de resoluções além da alta definição (UHD - *Ultra High Definition*) estão criando uma demanda de eficiência de codificação superior a eficiência alcançada pelo H.264/AVC (ALVAREZ-MESA *et al.*, 2012).

A necessidade é ainda maior quando altas resoluções de vídeo são usadas para sistemas 3D multivisão. Além disso, o tráfego causado por aplicações de vídeo destinadas a dispositivos móveis e tablets, bem como as necessidades de transmissão de serviços de vídeo sob demanda, estão impondo grandes desafios às redes de transmissão. Um aumento do desejo por maior qualidade e maiores resoluções também surge em usuários de aplicações móveis (ZENG ABDUL REHMAN *et al.*, 2013).

O HEVC foi desenvolvido essecialmente para atender a todas as aplicações do padrão H.264/AVC e para focar em duas áreas chaves: Altas resoluções de vídeo e o melhor aproveitamento de recursos de arquiteturas de processamento paralelo. A sintaxe do HEVC é genérica e deve também ser adequada a outras aplicações que não foram mencionadas no texto (PODDER *et al.*, 2015).

Para garantir o auxílio à indústria no processo de adoção do padrão, o esforço de padronização não inclui apenas o desenvolvimento de um documento textual de especificação, mas também de um codificador de referência para servir de exemplo de como o vídeo é codificado. O *software* de referência foi usado como a ferramenta de pesquisa para o trabalho interno do comitê durante o processo de padronização, e pode ser usado como uma ferramenta de pesquisa geral sobre o HEVC ou como base para novos produtos.

O Artigo (OHM *et al.*, 2012) apresenta as principais modificações do HEVC em relação ao seus antecessores e avalia a qualidade do vídeo usando a métrica objetiva PSNR e métricas subjetivas. A métrica PSNR, no caso do HEVC, além de ser usada como critério de avaliação

do vídeo produzido é também usada como critério de distorção para a otimização dentro do processo de codificação.

Há razões para o uso dessa métrica com esse intuito. A fórmula para seu cálculo é de fácil implementação e rápida execução. Apesar de sua popularidade, a PSNR tem apenas uma relação aproximada com a qualidade do vídeo percebido por observadores humanos. Isso ocorre porque ela se baseia em uma comparação *byte* a *byte* dos dados, sem considerar o que eles realmente representam. Assim, a medida fornecida por essa métrica não apresenta uma boa correlação com a qualidade realmente percebida.

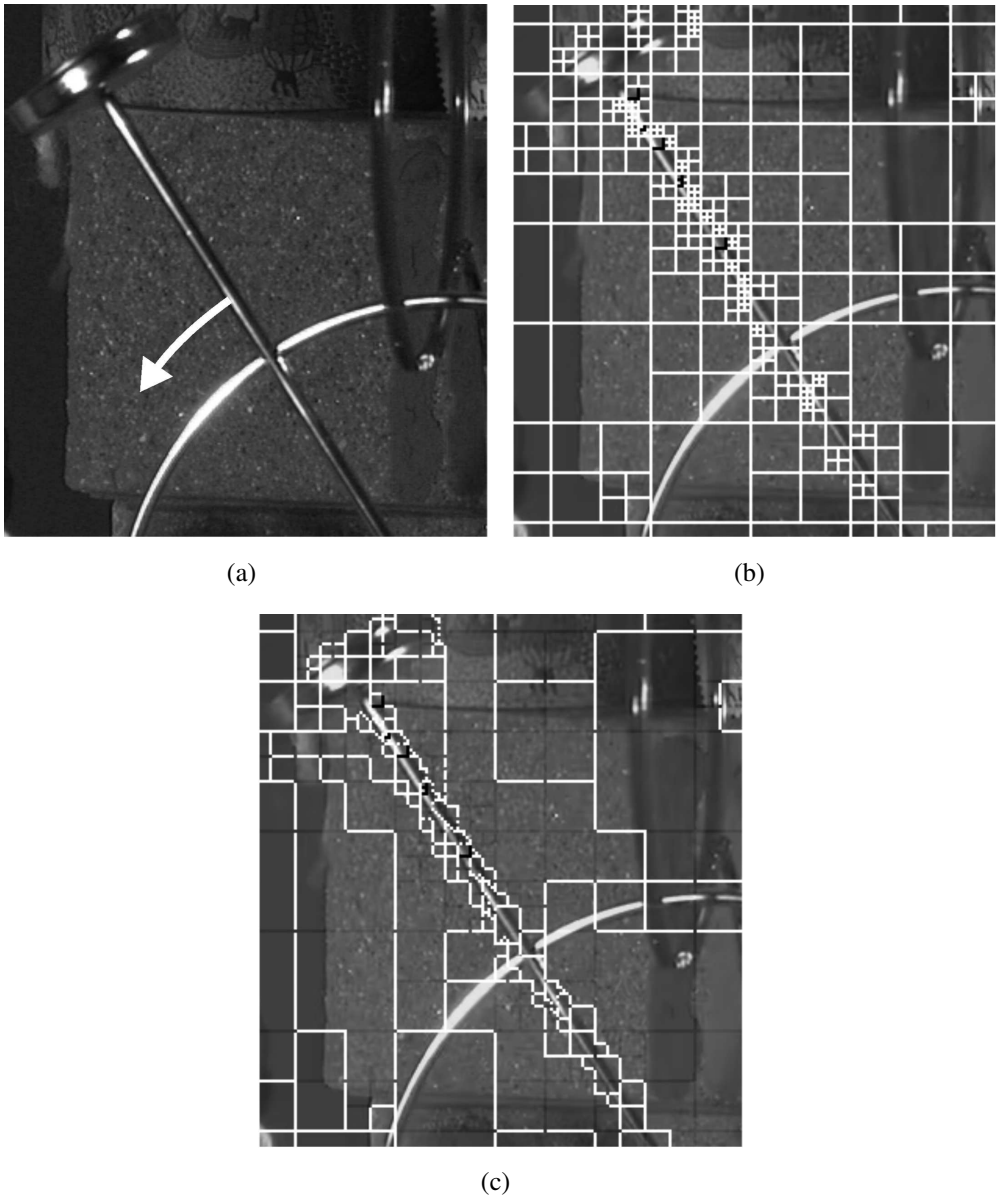
### 3.2.3 A Nova Estrutura de Particionamento de Blocos do Padrão HEVC

A principal motivação para a utilização do particionamento baseado em blocos em compressão de vídeo ou imagens é a possibilidade de codificar cada bloco com uma configuração específica escolhida dentro de um conjunto de parâmetros pré-definidos, considerando que, em geral, um modelo único não consegue mapear eficientemente as propriedades de uma imagem completa. A estrutura de dados Quadtree permite o particionamento de uma imagem em blocos de tamanho variável e é, portanto, um *framework* adequado para otimizar a contrapartida entre precisão do modelo (dado por uma distorção $D$) e o custo de codificação do modelo, tipicamente medida em taxa de *bits* R. Algoritmos de árvore que otimizam o funcional langrangeano $D + \lambda \times R$ foram desenvolvidos em (OHM *et al.*, 2010) e (VANNE *et al.*, 2012), motivando o projeto de algoritmos de compressão. O codificador/decodificador HEVC utiliza a abordagem de particionamento Quadtree com elementos sintáticos que armazenam a informação de subdivisão de blocos de diferentes tipos. Esse elementos sintáticos servem tanto para a subdivisão dos blocos como parâmetros para as etapas de transformação e predição do codificador.

No entanto, o conceito de codificação de imagens e vídeos particionados pela abordagem Quadtree é intrinsecamente relacionado a algumas contrapartidas, como as analisadas em (WIEGAND *et al.*, 2012). Por exemplo, a decomposição sistemática de todos os blocos em quatro blocos filhos não permite a representação conjunta de blocos filhos que pertençam a dois diferentes blocos-pai. Também, se um dado bloco é dividido em quatro blocos filhos, todos os filhos são tipicamente codificados separadamente, mesmo se dois ou três deles compartilhem os mesmos parâmetros de codificação. No entanto, as propriedades sub-ótimas subdivisão Quadtree inicial pode ser substancialmente aprimorada pela junção de nós (blocos) pertencentes a blocos-pai pontencialmente diferentes (ALVAREZ-MESA *et al.*, 2012).

Um exemplo para ilustrar o potencial da junção de blocos é mostrado na Figura 3.1. Essa cena particular exemplifica as contrapartidas do particionamento baseado no Quadtree discutido anteriormente. A Figura 3.1(a) exibe um segmento de um quadro da sequência de testes *Cactus* com uma seta indicando o sentido do movimento de uma haste de metal. A Figura 3.1(b) revela que uma divisão de blocos otimizada em termos da razão taxa/distorção força a segmentação Quadtree a dividir as regiões desnecessariamente. Foi observado, considerando uma gama de

sequências de vídeo e condições de codificação, que para esses blocos, que são colaterais a blocos previamente codificados com exatamente os mesmos parâmetros de movimento, que em 40% dos casos podem ser usados os mesmos parâmetros do bloco anterior. Para esse experimento, foi usado uma versão do codificador de referência HM sem a estimação para o algoritmo de fusão de blocos. O resultado foi uma indicação de uma super segmentação devido ao algoritmo de particionamento Quadtree e também indicou o potencial de codificação conjunta de blocos vizinhos com parâmetros de movimento idênticos (POURAZAD *et al.*, 2012).



(a) (b)

(c)

**Figura 3.1** (a) Segmento de uma quadro do vídeo *Cactus* contendo um objeto movendo-se na direção apontada pelo seta. (b) Segmentação resultante do Quadtree. (c) Particionamento do quadro após fusão de blocos Quadtree (WIEGAND *et al.*, 2012).

A Figura 3.1(c) mostra o mesmo particionamento como na Figura 3.1(b), mas com as fronteiras removidas entre blocos distintos com parâmetros de movimento idênticos. Isso revela

que a formação de regiões com as mesmas informações de movimento captam melhor os diferentes tipos de movimento na cena, proporcionando um diminuição no custo geral do processo de codificação. Idealmente, o limite de cada região deve coincidir com as descontinuidades de movimento de um determinado sinal de vídeo. Usando o particionamento de blocos baseado no algoritmo de particionamento em árvore combinado com fusão, a imagem pode ser subdividida em blocos cada vez mais pequenos, aproximando, assim, o limite de movimento e minimizando o tamanho das partições.

Deve notar-se que um esquema de fusão de blocos espacialmente vizinhos é conceitualmente semelhante aos modos de predição espacial como, por exemplo, o modo espacial direto da norma H.264/AVC [16]. Este modo também tenta reduzir o custo de codificação usando redundâncias de parâmetros de movimento em blocos vizinhos. No entanto, as melhorias em relação ao H.264/AVC de mostradas nos trabalhos anteriores (VANNE *et al.*, 2012) e (OHM *et al.*, 2012) sugerem que o conceito de fusão é superior em explorar estas redundâncias. Essa argumentação será testada nesse trabalho avaliando se essa nova organização espacial, que tende a ser mais alinhada com a experiência de visão, poderá produzir melhores resultados com técnicas de avaliação objetivas que tentam mapear aspectos subjetivos da visão.

### 3.2.4 O Algoritmo de Fusão de Blocos

O padrão HEVC especifica um algoritmo de fusão de blocos para diminuir a redundância de parâmetros em uma região do quadro. Esse algoritmo gera um conjunto único de parâmetros de movimentação para uma região contígua de blocos da imagem.

### 3.2.5 A Nova Estrutura de Blocos

O núcleo da camada de codificação nos padrões anteriores era o macrobloco, contendo um bloco de 16×16 de *pixels* de luminância e, no caso usual de uma amostragem de cor no formato 4:2:0, dois blocos 8×8 de amostras de crominância. Ao passo que, o padrão HEVC, tem como estrutura análoga ao macrobloco o formato conhecido como unidade de codificação em arvóre, que será chamado, ao longo desse texto, CTU (*Coding Tree Unit*). O CTU tem um tamanho selecionado em tempo real pelo codificador e pode ser maior ou menor que um macrobloco tradicional. Um CTU consiste em um bloco de codificação em árvore (CTB - *Coding Tree Block* de informações de luminância e os CTBs de crominância correspondentes e os elementos sintáticos. O tamanho $L \times L$ de um CTB de luminância pode ter $L$ igual a 16, 32 ou 64 amostras, com os tamanhos maiores tipicamente proporcionando melhor compressão. Dessa forma o HEVC suporta um particionamento dos CTBs em blocos menores usando uma estrutura de árvore baseada no algoritmo Quadtree.

Cada CTB é a raiz de uma árvore de codificação e dividido em blocos de codificação (CB). O seu tamanho pode ser escolhido de forma adaptativa usando um algoritmo de particionamento em árvore, chamado Quadtree, com as folhas representando os CBs [7]. Cada CB é

uma raiz de uma árvore de previsão e uma árvore de transformação. Cada árvore de previsão tem apenas um nível e descreve com um bloco de codificação (*Coding Block* – CB) ainda pode ser dividido nos chamados blocos de predição. em que, para cada um, diferentes padrões de predição são especificados (POURAZAD *et al.*, 2012).

O padrão HEVC também suporta o particionamento assimétrico das unidades de codificação nos formatos 64×16, 64×48, 16×64, 48×64, 32×8, 8×32, 24×32, 16×4, 16×12, 4×16 e 12×16. A complexidade algorítmica e a estrutura de dados do HEVC é, pelo menos, quatro vezes maior do que a do H.264/AVC. Dessa forma, diversos dispositivos eletrônicos com capacidade de processamento limitado e reduzida de autonomia de tempo de uso, não suportaram aplicações que usem codificação e/ou decodificação HEVC. No entanto, para selecionar um modo de predição, o codificador chega exaustivamente a função custo de Lagrange para todos os modos de predição em todos os níveis de profundidade da codificação.

### 3.2.6 Unidades de Codificação e Blocos de Codificação

A sintaxe Quadtree do CTU especifica o tamanho e as posições de seus blocos de codificação de luminância e crominância. A raiz da árvore de segmentação é associada com o CTU. Dessa forma, o tamanho do CTB de luminância é o tamanho máximo suportado por um CB de luminância. A divisão de um CTU em CBs de luminância e crominância é sinalizado conjuntamente. Um CB de luminância e dois CBs de crominância em conjunto com a sintaxe associada formam uma Unidade de codificação (*Coding Unit* – CU). Um CTB pode conter apenas um CU ou ser dividido para formar CUs múltiplos, e cada CU tem um particionamento associado em unidades de predição (*Prediction Unit* – PU) e uma árvore de unidades de transformação (*Transform Unit* – TU).

### 3.2.7 Unidades de Predição e Blocos de Predição

A decisão de codificar a região de uma imagem usando predição inter ou intra é feita na unidade de codificação. Uma estrutura de particionamento das unidades de predição tem sua própria raiz. Dependendo decisão de predição básica, os CBs de luminância e crominância ainda podem ser divididos e preditos a partir de blocos de predição de crominância e luminância. O HEVC suporta tamanho do blocos de predição variando de 64×64 até 4×4.

### 3.2.8 Unidades de Transformação e Blocos de Transformação

O resídios gerados na etapa de predição são codificados usando entidades conhecidas como blocos de transformação (*Transformation Blocks* – TB). O tamanho de um TB de luminância pode ser idêntico ao tamanho de um CB de luminância, mas pode ser dividido em TBs menores. O mesmo acontece para os TBs de crominância. Uma DCT (*Discrete Cosine Transform*) é utilizada para o cálculo das tranformadas para os TBs de tamanho 8×8, 16×16 e

32×32. Para o tamanho 4×4, uma transformada inteira derivada da Transformada Discreta de Senos é especificada.

### 3.2.9 A Estimação de Movimento

A implementação de referência do codificador HEVC, o codificador HM, inclui um algoritmo de estimação de movimento rápido e eficiente baseado no algoritmo de busca preditiva local aprimorada (EPZS – (*Enhanced Predictive Zonal Search*)). O EPZS faz uso de uma combinação de algoritmos de predição de vetores de movimento e padrões de busca para reduzir significativamente o número de vetores de movimento candidatos que são testados durante a etapa de estimação. O EPZS é usado na estimação de movimento unidirecional, enquanto que na etapa bidirecional uma busca exaustiva é realizada em uma pequena janela em torno do ponto previamente encontrado na etapa unidirecional (ZUPANCIC *et al.*, 2015a). Recentemente, foi proposto um método de terminação da estimação de movimento chamado de MET (*Multiple Early Termination*). Esse algoritmo executa um padrão de busca pré-definido em volta do pontos iniciais determinados pelo EPZS e caso encontre um resultado satisfatório, o processo de estimação de movimento é abortado nos seus primeiros estágios (ZUPANCIC *et al.*, 2015b).

Semelhantemente ao H.264/AVC, múltiplos quadros de referência são usados. Para cada PB, um ou dois vetores de movimento podem ser transmitidos, resultados da predição unidirecional e bidirecional, respectivamente. Uma operação de escalonamento e deslocamento, conhecida por predição ponderada, pode ser aplicada ao vídeo resultante do processo de predição/compensação de movimento.

### 3.2.10 Predição dos Vetores de Movimento

Uma nova técnica chamada *Advanced motion vector prediction* (AMVP) (SZE MADHUKAR BUDAGAVI, 2014a) foi usada no HEVC. Essa técnica consiste em incluir derivações dos vetores de movimento candidatos mais prováveis com base em informações de PBs (*Prediction Blocks*) adjacentes e do quadro de referência para uma fusão de vetores de movimento que herda os vetores de PBs vizinhos espaciais ou temporais. E ainda, em relação ao H264/AVC, novos aprimoramentos foram feitos para as técnicas de inferência de salto de vetor e inferência direta de movimento.

A fusão de vetores de movimento propõe a criação de uma lista com a informação de movimento de unidades de predição (PU –*Prediction Unit*) vizinhas. As PUs candidatas estão posicionadas espacialmente ou temporalmente perto da PU corrente. O codificador sinaliza qual candidata da lista vai ser usada e a informação de movimento da PU será copiada da candidata selecionada (CORPORATION; WAY, 2010). Importante notar que a fusão de movimento evita a necessidade de processar um vetor de movimento para a PU; no seu lugar, apenas o índice de uma PU candidata na lista de fusão de movimento é transmitida.

No modo Skip do HEVC, o codificador também codifica o índice de um candidato de fusão de movimento, e os parâmetros de movimento para a PU corrente são copiados do candidato selecionado. Isso permite que áreas do quadro que tenham pouca movimentação, ou mudanças, entre os quadros ou tenha movimento constante sejam codificadas usando uma quantidade bem menor de *bits*.

### 3.2.11 Predição Intra

As amostras de (*pixels*) decodificadas posicionadas nas bordas de blocos são usadas como informação de referência para a predição espacial em regiões onde a predição inter não é realizada. A predição intra no HEVC suporta 33 modos direcionais, além do modo planar e do modo DC. Cada direção indica o sentido em que a busca espacial por blocos similares ocorre. O H.264/AVC especifica o suporte a oito modos direcionais. A Figura 3.2 ilustra esse conceito. Os modos de predição intra selecionados são codificados derivando o modo de mais provável baseando-se nos PBs vizinhos previamente decodificados.

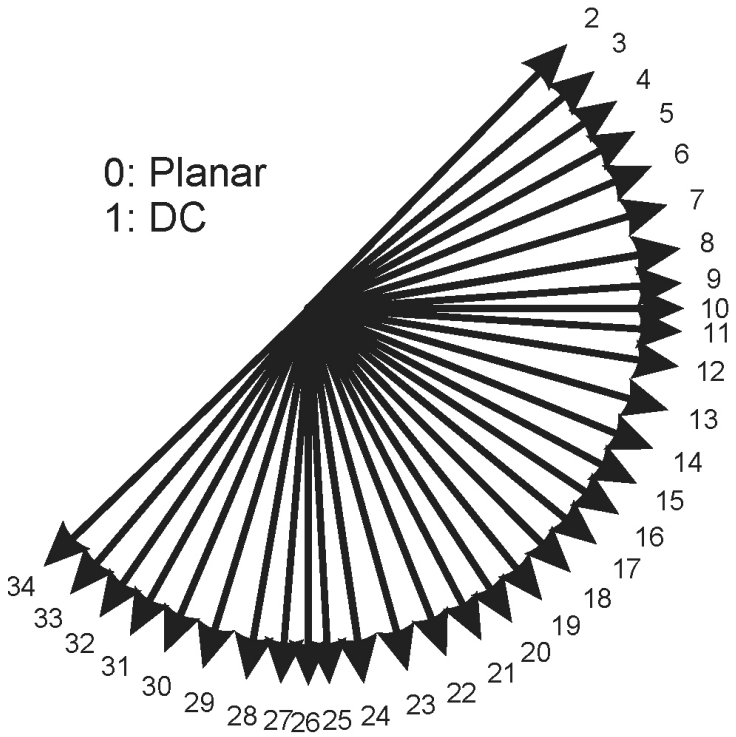



**Figura 3.2** Modos de predição direcional/espacial na predição intra.

### 3.2.12 Controle de Quantização

Como no H.264/AVC, a técnica de quantização com reconstrução uniforme também é usada com matrizes escaleres de quantização suportadas para os vários tamanhos de TBs (*Transform Blocks*).

### 3.2.13 Codificação de Entropia

O HEVC especifica como codificador de entropia a técnica CABAC (*Context Adaptive Binary Arithmetic Coding*) (SZE MADHUKAR BUDAGAVI, 2014a). A implementação dessa técnica passou por várias otimizações quando comparado à última versão do código de referência do H.264/AVC. Destacam-se o aumento da capacidade de taxa de compressão, devido ao foco em arquiteturas de processamento paralelo e o aumento na eficiência de compressão visando reduzir a quantidade de memória alocada pelos modelos contextuais.

### 3.2.14 Filtro de Desblocagem

A especificação do HEVC manteve o filtro de desblocagem que havia sido especificado para a arquitetura do H.264/AVC (SZE MADHUKAR BUDAGAVI, 2014b). No entanto, algumas melhorias foram implementadas, como a simplificação do projeto na parte responsável pelos processo de tomada de decisão. Além disso, essa nova implementação tornou-se mais adequada para plataforma com processamento paralelo.

Enquanto que no H.264/AVC cada borda de um *grid* $4\times4$ deve ser filtrado, o padrão HEVC limita a filtragem aos *pixels* das bordas de um *grid* $8\times8$. Isso imediatamente reduz pela metade o número de modos de filtragem que precisam ser calculados e o número de amostras que precisam ser filtradas. A ordem em que as bordas são processadas também foi modificada para aprimorar o processamento paralelo. Um quadro pode ser segmentado em blocos $8\times8$ que podem ser processados em paralelo, uma vez que apenas as bordas internas desses blocos precisam ser filtradas. A posição desses blocos está ilustrada na Figura 3.3 que ilustra o alinhamento (linha pontilhada) dos blocos $8\times8$ nos quais o filtro de desblocagem pode ser aplicado independentemente. As linhas sólidas representam as bordas dos CTBs (*Coding Tree Blocks*). Alguns desses blocos ultrapassam os limites do CTBs e dos *slices*. Essa funcionalidade possibilita filtrar as bordas dos *slices* (OHM; SULLIVAN, 2013).

Nesse processo de filtragem, as bordas verticais são filtradas antes de bordas horizontais. Consequentemente, as amostras modificadas resultantes da filtragem das bordas verticais são usadas na filtragem das bordas horizontais. Isto possibilita diferentes implementações paralelas. Em uma das abordagens, todas as bordas verticais são filtradas em paralelo, em seguida, as bordas horizontais são filtradas em paralelo. Outra opção seria permitir o processamento paralelo simultâneo das bordas verticais e horizontais, em que o processo de filtragem da aresta horizontal é retardado de uma maneira tal que as amostras a serem filtrados já foram processadas pelo filtro de borda vertical.

### 3.2.15 Deslocamento Adaptativo de Amostras

Comparando com o H.264/AVC, no qual apenas a filtragem de desblocagem é aplicada no processo de recuperação da imagem de referência dentro do codificador, o padrão HEVC es-

pecifica um novo filtro chamado deslocamento adaptativo de amostras (*Sample Adaptive Offset* – SAO). Esse filtro representa um estágio adicional que aumenta a complexidade computacional do codificador (OHM; SULLIVAN, 2013).

A técnica de deslocamento adaptativo de amostras introduz um mapeameneto não-linear de amplitude dentro do processo de predição inter depois do filtro de desblocagem. Essa técnica tem como meta uma melhor reconstrução das amplitudes do sinal original usando uma tabela de consulta (*look-up table*) que é descrita por alguns parâmetros adicionais que podem ser determinados por análise do histograma no lado do codificador.

O índice para a tabela de consulta pode ser calculado de acordo com um dos dois modos utilizados. Para o modo de deslocamento de banda (BO – *Band Offset*) os valores das amostras são quantizados de acordo com o respectivo índice da tabela. Assim, todas as amostras que encontram-se dentro de uma faixa de intervalo de valores são processadas usando o mesmo deslocamento. O outro modo, chamado modo de deslocamento de borda (EO – *Edge Offset*), exige mais operações, uma vez que calcula o índice com base nas diferenças entre as amostras do bloco corrente e de duas amostras vizinhas. O modo EO usa um de quatro padrões unidimensionais para classificar os *pixels* baseado na direção de sua borda mais próxima, como ilustrado pela Figura 3.4. Cada *pixel* pode ser classificado como pico (se seu valor é maior do que o de seus vizinhos), vale (se seu valor é menor do que seus vizinhos), borda (se seu valor é igual ao valor de algum vizinho) ou nenhum desses. Quatro valores de deslocamentos serão calculados para cada uma dessas categorias. O codificador pode escolher aplicar qualquer uma das duas técnicas em diferentes regiões de uma mesma imagem. Também é possível sinalizar que nenhuma das opções foi aplicada.

Embora as operações sejam simples, o SAO (*Sample Adaptative Offset*)representa uma carga adicional uma vez que pode necessitar ou uma passagem de decodificação adicional, ou um aumento dos *buffers* de memória.

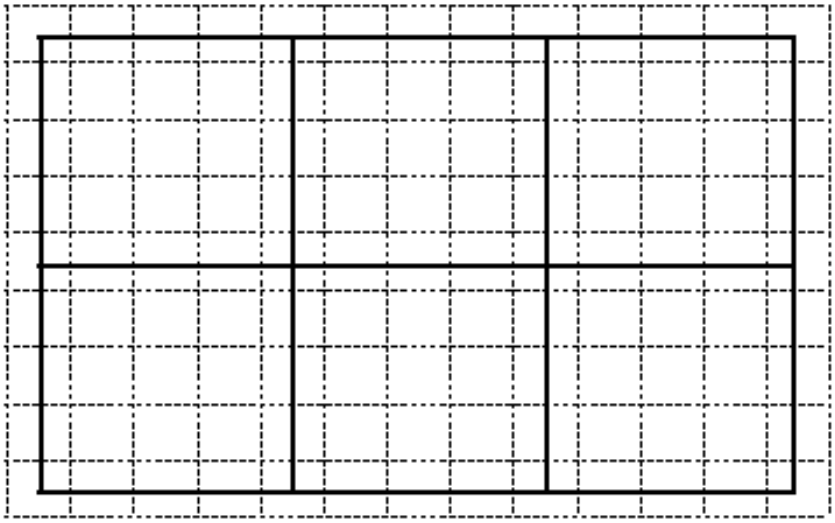


**Figura 3.3** Alinhamento para o filtro de desblocagem.

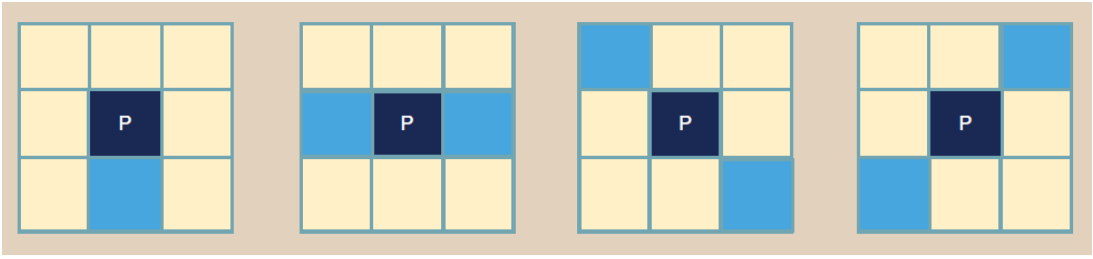



**Figura 3.4** Padrões usados no modo EO do SAO.

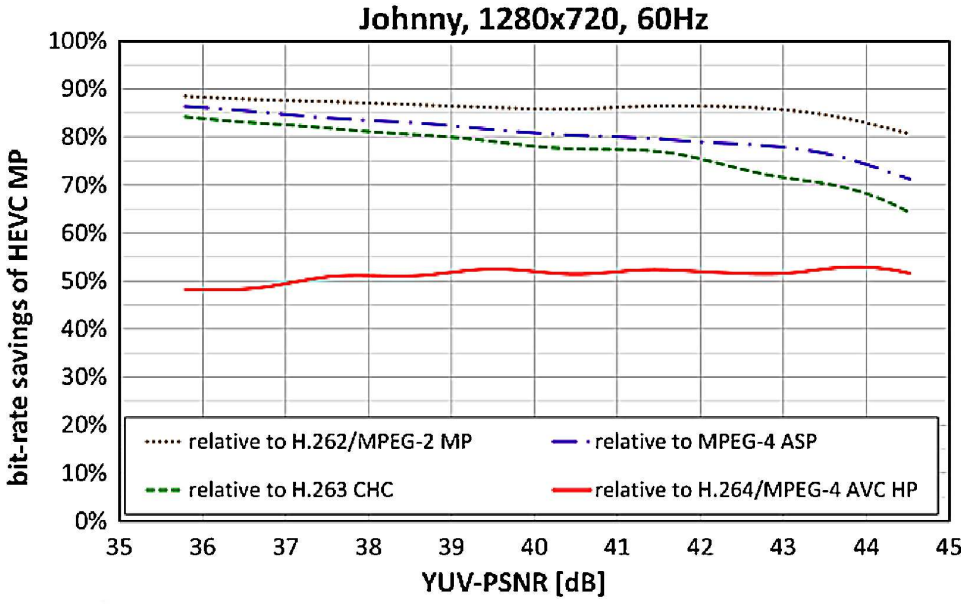



**Figura 3.5** Ganho de taxa de *bits* em relação a codificadores anteriores.

## 3.3 O Codificador de Referência HM

O principal propósito do codificador de referência HM é prover um implementação de referência comum de um codificador HEVC que seja útil como uma plataforma de testes para avaliação de tecnologias de codificação e para referência de desenvolvimentos independentes de codificadores e decodificadores em conformidade com o padrão HEVC. Escrito em C++, o codificador HM não foi projetado visando ser um código de produção, pronto para ser embarcados em produtos de mercado. Ainda há bastante o que ser feito em termos de otimização de desempenho. Ainda é significativamente lento e para a otimização do desenvolvimento e testes, *clusters* ou super-computadores são recomendados. Mesmo que a velocidade do codificador HM seja lenta para a maioria das situações, alguns aprimoramentos de tempo foram feitos durante o seu desenvolvimento. Enquanto mais de 100 horas foram necessárias para codificar um único vídeo HD de 10 segundos na primeira versão do codificador HM, após várias melhorias, chegou-se a menos de 20 horas para a mesma tarefa. Entre os fatores que contribuem para essa lentidão do codificador o que provavelmente mais influencia é o processo de otimização em tempo de execução da razão taxa/distorção. Essa otimização é aplicada durante o processo de quantização (VIITANEN *et al.*, 2013).

As condições de testes especificadas pelo comitê JCT-VC definem um conjunto de configurações para o codificador para o uso em experimentos de comparação. Essas configurações são (TAN *et al.*, 2015):

- *all intra* (AI), em que todos os quadros são codificados usando *slices* I;
- *random access* (RA), em que é usada uma reordenação dos quadros com uma escolha aleatória a cada segundo. Essa configuração emula o que deve acontecer em um ambiente de transmissão terrestre;
- Pequeno atraso com *slices* B (LB), em que nenhuma reordenação é usada e apenas o primeiro quadro é codificado usando *slices* I. Essa configuração emula o que deve acontecer em transmissões de vídeo-conferências.

Enquanto que a maioria das funcionalidades do HEVC são utilizadas nessas três configurações citadas anteriormente, algumas outras não são, incluindo a predição ponderada e o escalonamento das matrizes de quantização. As ferramentas de paralelismo mais avançadas também foram desabilitadas.

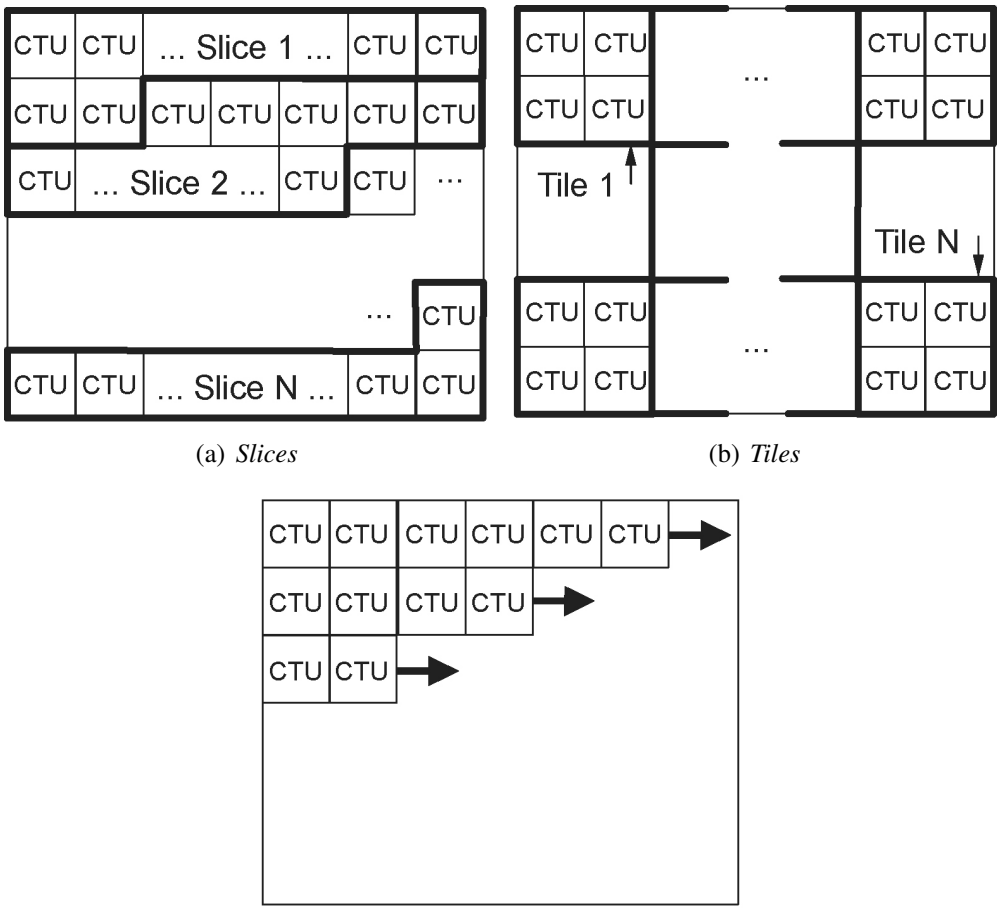
## 3.4 O Decodificador de Referência HM

Semelhante ao codificador de referência, o decodificador HM é um exemplo de implementação que visa exatidão e integridade. Ele é executado em um única *thread*, e não utiliza

nenhuma técnica de paralelização. É esperado que vários componentes do decodificador sejam aprimorados para alcançar um desempenho em tempo real (VIITANEN *et al.*, 2013).

## 3.5 Oportunidades de Paralelização Aplicadas pelo Padrão HEVC

Os codecs (Codificador/Decodificador) de vídeo anteriores, e.g. H.264/AVC, usaram estrátegias de paralelização ao nível de *slice* ou de bloco. No caso de paralelismo a nível de *slices*, um quadro é dividido em diversos *slices*, os quais são completamente independentes entre si. *Threads* múltiplas podem ser usadas para processar em paralelo os *slices* de um quadro, aumentando a vazão e, ao mesmo tempo, diminuindo a latência. Além dos *slices*, o padrão HEVC define mais duas técnicas de paralelização: *Tiles* e o processamento paralelo em frentes de onda (WPP) (ALVAREZ-MESA *et al.*, 2012). A Figura 3.6 ilustra como as unidades de codificação em árvore (CTU) são agrupadas para serem processados paralelamente.




(a) *Slices* (b) *Tiles*

(c) WPP

**Figura 3.6** Técnicas de paralelização de processamento.

O aumento da quantidade de *slices* em um quadro reduz a eficiência de codificação significantemente devido a três motivos:

- A codificação de entropia é menos eficiente devido às quebras no processo de treinamento dos modelos contextuais e a impossibilidade de ultrapassar as fronteiras dos *slices* para aumento do contexto selecionado.
- No estágio de predição, os *pixels* de *slices* vizinhos não podem ser utilizados.
- Para cada *slice* um cabeçalho adicional e código de ínicio precisa ser adicionado no fluxo de *bits* final.

Paralelismo a nível de bloco não depende de haver múltiplos *slices* em um quadro e não possuem as perdas de codificação associadas ao paralelismo a nível de *slice*. No seu lugar, os blocos de codificação (LCU – *Largest Coding Unit*) dentro de um quadro podem ser reconstruídos em paralelo usando uma abordagem de frente de onda para satisfazer as dependências de filtragem e predição. No entanto, a decodificação de entropia não pode ser paralelizada a nível dos blocos.

### 3.5.1 A Ferramenta *Entropy Slices*

A especificação HEVC introduziu uma nova ferramenta de codificação chamada de *Entropy Slices* (OHM; SULLIVAN, 2013), que diferentemente dos *slices* comuns, foi desenvolvida com fins de privilegiar a paralelização de processamento em detrimento da resiliência a erros (ALVAREZ-MESA *et al.*, 2012). Os modelos contextuais são inicializados no ínicio de cada *slice*, tanto para os comuns como para os *entropy slices*. A principal diferença é que, nas fases de filtragem e reconstrução, é permitido usar informações de blocos vizinhos nas bordas dos *Entropy Slices*. Outra diferença é que o cabeçalho dos *entropy slices* é menor do que o dos *slices* comuns, porque as informações de cabeçalho são enviadas somente no primeiro cabeçalho de cada *slice* de um quadro.

Com o uso dessa ferramenta, multiplas *threads* podem decodificar o mesmo quadro ao mesmo tempo, com o benefício de diminuir a latência de quadros em projetos que usam paralelismo a nível de bloco. Como a etapa de decodificação de entropia é desacoplada das etapas de reconstrução e filtragem, em geral, é necessário uma área de memória de tamanho significativo para armazenar todos os dados após a etapa de decodificação de entropia, mas algumas implementações já conseguem dispensar o uso dessas áreas de memória (ALVAREZ-MESA *et al.*, 2012).

### 3.5.2 A Ferramenta *Tiles*

A opção de dividir uma imagem em regiões retangulares chamadas *Tiles* foi especificada pela primeira vez no padrão HEVC. A finalidade principal é aumentar a capacidade de

processamento paralelo, em vez de fornecer resiliência a erro. *Tiles* são regiões independentemente decodificáveis da imagem, contém informações compartilhadas de cabeçalho. *Tiles* podem ser adicionalmente utilizadas com a finalidade de acesso espacial aleatório para regiões das imagens de vídeo. Uma configuração típica de uma *tile* de imagem consiste de segmentar a imagem em regiões retangulares com aproximadamente o mesmo número de CTUs (*Coding Tree Unit*) em cada azulejo. Uso de *Tiles* possibilita o paralelismo em um nível mais grosseiro de granularidade, e nenhuma sincronização sofisticada de *threads* é necessária para seu uso.

### 3.5.3 Processamento Paralelo em Frentes de Onda

Quando o processamento paralelo (WPP – *Wavefront Parallel Processing*) em frente de onda é habilitado, um *slice* é dividido em linhas de CTUs. A primeira linha é processada normalmente, enquanto que a segunda linha apenas pode ser processada depois de que apenas dois CTUs da primeira linha já tenham sido processados. O processamento da terceira linha só é iniciado quando após o processamento de dois CTUs da segunda linha, e assim por diante. Os modelos contextuais do codificador de entropia para cada linha são inferidos a partir dos modelos determinados das linhas anteriores. A técnica WPP, em geral, possibilita um melhor desempenho de compressão do que a técnica *Tiles* e evita alguns artefatos visuais que podem ser inserido pelo uso de *Tiles* (OHM; SULLIVAN, 2013).

## 3.6 Estimação de Movimento com o Algoritmo TZSearch

O Algoritmo TZSearch (PAN *et al.*, 2013) é um dos algoritmos de estimação de movimento usados no processo de exploração de redundância temporal dos codificadores HEVC JM e JMVC (usado para sequências multi-visões). Seu principal ponto forte é a redução da complexidade computacional do processo de estimação de movimento em comparação com os métodos usados no H.264 sem afetar a taxa de *bits* e a PSNR medida significantemente. O processo do algoritmo é descrito nos seguintes passos (NALLURI *et al.*, 2012):

1. No ínicio, um ponto inicial deve ser determinado como ponto central para o próximo passo. O algoritmo calcula a soma das diferenças absolutas (SAD – *Sum of Absolute Differences*) de cinco pontos com vetores de movimento preditos diferentes (zero, média, esquerda, cima e cima-direita) no quadro de referência. O ponto inicial escolhido é que possuir menor SAD.

2. Na busca inicial, padrão de busca de diamante de oitos pontos ou o quadrado de oito pontos é aplicado com passos de busca diferentes. Se o intervalo de busca for igual a 64, o passo de busca varia de 1 a 64 em múltiplos de 2. O ponto com menor SAD é o ponto central de uma provável próxima busca dentro do ciclo de refinamento da posição do vetor de movimento. O passo de busca é armazenado como a menor distância.

3. Se a menor distância for igual a 1, o padrão de busca de dois pontos é utilizado para calcular a SAD dos dois pontos próximos ao centro e a menor distância é configurada como 0. Se a menor distância for maior do que o parâmetro que define o valor mínimo, o padrão de busca em trama é aplicado para encontrar o ponto com menor SAD que será o centro da busca do próximo passo.

4. Se a menor distância for maior do que 0, o refinamento em trama ou o refinamento estrela até que a menor distância seja 0. Cada uma desses refinamentos incluem os padrões de busca diamante de oito pontos, quadrado de oito pontos e o padrão de busca de dois pontos. Quando a menor distância for zero, o centro da busca é o ponto ótimo e o algoritmo de estimação de movimento TZSearch é finalizado.

### 3.6.1 Métricas de Distorção de Blocos

Os algoritmos de estimação de movimento baseiam-se na comparação entre blocos de imagem no quadro corrente e em quadros anteriores ou posteriores. Essa comparação é realizada atráves de métricas de distorção de blocos. Para a explicação das métricas apresentadas nesse capítulo, um bloco de tamanho $N \times N$ é considerado. O valor do *pixel* na coordenada $(n_1, n_2)$ no quadro $k$, $k$ sendo o índice do quadro no vídeo, é dado por $S(n_1, n_2, k)$ em que $0 \leq n_1$, $n_2 \leq N-1$. O quadro $k$ representa o quadro corrente, assim como o bloco de *pixels* descrito acima representa o bloco corrente. Nas seções 3.6.1, 3.6.1, 3.6.1 são descritas três medidas de distorção de bloco empregadas em codificadores de vídeo.

#### Erro Médio Quadrático

Considerando $k$ como os quadros de referência passados, no processo de estimação de movimento, o erro médio quadrático (*Mean Square Error* – MSE ) para um bloco de $N \times N$ *pixels* é dado por (KAMOLRAT *et al.*, 2009) (BARJATYA; MEMBER, 2004)

$$\mathrm{MSE}(i,j,k) = \frac{1}{N^2} \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} [S(n_1,n_2,k) - S(n_1+i, n_2+j, k)]^2. \tag{3.1}$$

O significado físico da Equação 3.1 deve ser bem compreendido. Considera-se um bloco de pixels de tamanho $N \times N$ no quadro de referência a um deslocamento de $(i, j)$, em que $i$ e $j$ são números inteiros em relação à posição do bloco candidato no quadro corrente.

O erro médio médio quadrático é calculado para cada posição $(i, j)$ dentro de um intervalo de pesquisa especificado na imagem de referência, e o deslocamento, que dá o menor valor de MSE é designado como vetor de deslocamento, ou vetor de movimento, dado por

$$[d_1, d_2] = \min_{i,j} [\mathrm{MSE}(i,j)]. \tag{3.2}$$

O cálculo do erro médio quadrático requer o processamento de $N^2$ subtrações, $N^2$ multiplicações e $(N-1)^2$ adições para cada bloco candidato em cada posição da busca. Isto é computacionalmente dispendioso e um critério mais simples, como definido na Subseção 3.6.1 é muitas vezes preferido ao critério MSE.

### Diferença Absoluta Média

Como o critério MSE, a diferença média absoluta (*Mean Absolute Difference* – MAD) também faz com que os valores de erro sejam sempre positivos, mas em vez de somar as diferenças de quadrados, as diferenças absolutas é que serão somadas. A medida das diferença média absoluta, MAD, é definida como (MAHMOUD *et al.*, 2006).

$$\mathrm{MAD}(i,j) = \frac{1}{N^2} \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} |S(n_1,n_2,k) - S(n_1+i,n_2+j,k)| \,. \tag{3.3}$$

O vetor de movimento é determinado de forma semelhante à medida MSE, que usa a Fórmula 3.4, definada por

$$[d_1,d_2] = \min_{i,j}[\mathrm{MSE}(i,j)]. \tag{3.4}$$

O critério MAD requer o processamento de $N^2$ subtrações de valores absolutos e $N^2$ adições para cada bloco candidato em cada posição de busca determinada pelo algoritmo de estimação de movimento. A ausência de multiplicações faz desse critério uma opção computacionalmente menos dispendiosa, facilitando assim possíveis implementações em *hardware*.

### Correspondência na Contagem dos *Pixels*

Para este critério, os *pixels* de um bloco candidato $B$ são comparados com os *pixels* correspondentes do bloco com deslocamento $(i,j)$ no quadro de referência e aqueles que são menores que um limiar especificado são contados. A contagem para a comparação, e o deslocamento $(i,j)$ para os quais a contagem é máxima, corresponde às coordenadas do vetor de movimento e é expressa em um função binária $c(n_1,n_2) \forall (n_1,n_2) \in B$ definida em 3.5.

$$\mathrm{c}(n_1,n_2) = \begin{cases} 1, & \text{se } |S(n_1,n_2,k) - S(n_1+i,n_2+j,k-l)| \leq \theta \\ 0, & \text{caso contrário} \end{cases} \tag{3.5}$$

em que $\theta$ é um limiar pré-determinado. A correspodência na contagem de *pixels* para um deslocamento $(i,j)$ é dada pelo valor acumulado da Equação 3.5, definida por 3.6

$$\mathrm{MPC}(i,j) = \sum_{n_1=0}^{N-1} \mathrm{c}(n_1,n_2), \tag{3.6}$$

e as respectivas coordenadas finais do vetor de movimento são definidas como

$$[d_1, d_2] = \min_{i,j}[\text{MPC}(i, j)]. \tag{3.7}$$

## 3.7 Avaliação de Desempenho

O desempenho da compressão do HEVC foi significantemente aprimorado em relação aos resultados do H264/AVC. As avaliações mostradas em (OHM *et al.*, 2012) indicam que, para as versões iniciais do codificador de referência do HEVC (HM), a taxa de *bits* foi quase reduzida à metade em relação às taxas obtidas com o codificador de referência H.264/AVC (JM), considerando a qualidade visual subjetiva. Configurado com os parâmetros para o perfil LP (*Low Profile*), o codificador HM HE (*High Efficiency*) pode alcançar 50% de redução na taxa de *bits* de saída em relação ao H.264 (VANNE *et al.*, 2012).

A melhor forma de avaliar o desempenho de compressão é medir a qualidade subjetiva percebida por observadores humanos. O objetivo do estudo feito em (BARONCINI J. OHM, 2012) é quantificar o ganho de taxa de *bits* do HEVC em relação ao H.264/AVC, quando comparado com o uso de medidas de qualidade subjetiva similares. Nesse estudo o codificador HM 5.0 foi usado, que é não é tão eficiente quanto a Versão 9.2, que foi utilizada no início dos estudos desta tese.

A partir desses resultados, a redução da taxa de *bit* média aproximada sobre todas as seqüências com qualidade subjetiva equivalente foi calculado em 67% para as cinco sequências full HD, 49% para as quatro sequências WVGA, e 58% no total. Estas medidas iniciais indicam que o HEVC cumpre sua meta inicial em termos de ganhos de qualidade subjetiva. Embora o número de casos de teste tenha sido limitado, o fato de a mesma tendência poder ser observada para todas as diferentes sequências é um indicador da melhoria substancial na eficiência de compressão que é proporcionada pelo HEVC.

Apesar dessas avaliações subjetivas da qualidade do vídeo, como a MOS (*Mean Opinion Score*), tenderem a ser as mais confiáveis, elas possuem execução bastante complexa. No entanto, medidas de qualidade objetivas repetíveis e automáticas, como a relação sinal-ruído de pico (PSNR – *Peak Signal-to-Noise Ratio*), o índice de similaridade estrutural (SSIM – *Structural Similarity Index*) e o índice de qualidade percebida (PQI – *Perceived Quality Index*) são tipicamente usadas quando as medidas subjetivas são impraticáveis ou de dificil execução.

O resultado do estudo realizado em (OHM *et al.*, 2012) mostrou uma economia na taxa de *bit* de aproximadamente 40% em relação ao HEVC e 70% à 80% em relação ao MPEG-2 *Main Profile*. Esse estudo usou uma abordagem de comparação bem disciplinada em relação a escolha das ferramentas e parâmetros de codificação para garantir uma comparação justa entre todos os codificadores escolhidos. Além do ganho médio em eficiência de compressão, esse testes indicaram algumas outras importantes características em comparação com os outros padrões. São elas:

- Os benefícios em termos de qualidade subjetiva de vídeo pareceram maior do que o que foi sugerido pelos resultados da PSNR.
- Os ganhos em eficiência de compressão indicaram ser maiores para vídeos em alta definição do que para vídeos com resoluções pequenas.
- Para as configurações testadas, os ganhos foram maiores em vídeos definidos para comunicação em tempo real do que para a aplicações de entretenimento.
- Quanto menor a taxa de *bits*, maior é o ganho proporcional em qualidade percebida.

As avaliações de qualidade objetivas existentes usam a relação entre taxa de *bits* e distorção, medida com alguma métrica derivada da PSNR, como em (SEGALL *et al.*, ), em que os codificadores de referência HM (HEVC) e JM (H.264) são comparados com o uso da métrica Bjøntegaard PSNR.

### 3.7.1 A Métrica Bjøntegaard

Ao comparar duas curvas de taxa de distorção, por exemplo, a medição do desempenho para duas variações dos esquemas de codificação testados, um número que representa as economias globais de taxa de *bits* ou a diferença geral de qualidade torna-se uma informação valiosa. A métrica Bjøntegaard tornou-se uma ferramenta popular para avaliar a eficiência de codificação de um determinado codificador de vídeo em comparação com um codificador de referência ao longo de um intervalo de pontos de qualidade ou taxas de *bits* (BJONTEGAARD, 2001). Essas métricas, desenvolvidas por Gisle Bjøntegaard, são normalmente calculadas como uma diferença na taxa de *bits* ou uma diferença de qualidade com base nas curvas de interpolação dos pontos dos dados testados. Neste trabalho o foco é sobre a diferença de taxa de *bits*, expresso como uma percentagem da taxa de bit resultante do processo de codificação do codificador de referência. Esse valor é interpretado como uma economia na taxa de *bits* para um mesmo parâmetro de qualidade medido. Nessa tese, o parâmetro de qualidade utilizado foi a relação sinal-ruído de pico (PSNR) da informação de luminância.

A taxa delta de Bjøntegaard (BD-*Rate*) representa a média de ganhos de taxa de bit para a mesma qualidade de vídeo, medida pela MOS ou pela PSNR e é calculada entre duas curvas de taxa$\times$distorção. A diferença de economia de taxa de *bits* entre essas curvas para um mesmo nível de qualidade é definida na equação 3.8

$$\Delta R = \frac{R_B(D) - R_A(D)}{R_A(D)}, \tag{3.8}$$

em que $R_A(D)$ e $R_B(D)$ são as taxas de *bits* resultantes do codificador de referência e do codificador em teste, respectivamente, em determinado nível de qualidade/distorção $D$. Valores negativos de $\Delta R(D)$ representam um ganho da relação compressão-qualidade, enquanto que valores positivos representam uma perda nessa relação.

A métrica de Bjøntegaard usa uma escala logarítmica, então definindo-se $r = logR$, os ganhos de taxa de *bits* podem ser expressados por

$$\Delta R = 10^{r_B(D) - r_B(D)} - 1. \tag{3.9}$$

A proposta original assume a medição do desempenho da taxa de distorção de duas configurações de codificação com quatro valores diferentes do parâmetro de quantização diferentes. O conceito básico é a interpolação da curva entre os pontos de taxa de distorção que foram medidos no experimento e para integrar mais a diferença. O cálculo é realizado sobre a taxa logarítmica a fim de obter a diferença da taxa relativa. Uma vez que o PSNR é uma medida logarítmica, ela pode ser aplicada diretamente.

### 3.7.2 Outras Métricas Objetivas de Qualidade

#### A Métrica PSNR

A relação sinal-ruído de pico (PSNR – *Peak Signal-to-Noise Ratio*) é expressa definindo o erro médio quadrático *Mean Squared Error* – (MSE) em relação ao valor máximo possível da luminância. Considerando a MSE, definida em 3.1 e para um valor de *n-bits*, a PSNR é expressa em dB como

$$\text{PSNR} = 20\log_{10}\frac{2^n - 1}{\sqrt{\text{MSE}}}(\text{dB}). \tag{3.10}$$

#### O Índice de Similaridade Estrutural (SSIM)

O índice de similaridade estrutural (SSIM – Structural Similarity Index) baseia-se no conceito de que imagens naturais são altamente estruturadas (FARIAS, 2011). Em outras palavras, imagens possuem uma forte correlação entre si, o que resulta em informações de objetos na cena. Esse método se diferencia de outros métodos de avaliação objetiva por usar uma medida da distorção estrutural em vez do próprio erro. Se $x = \{x_i | i = 1, 2, \ldots, N\}$ representa o sinal original e $y = \{y_i | i = 1, 2, \ldots, N\}$ representa o sinal distorcido, em que $i$ é o valor do índice do *pixel*, o índice de similaridade estrutural (SSIM) pode ser calculado segundo (YASAKETHU *et al.*, 2009)

$$\text{SSIM} = \frac{(2 \cdot \bar{x} \cdot \bar{y} + C_1)(2 \cdot \sigma_{xy} + C_2)}{(\bar{x} + \bar{y}^2 + C_1)(\sigma_x^2 + \sigma_y^2) + C_2}. \tag{3.11}$$

Na Fórmula 3.11 $\bar{x}$, $\bar{y}$, $\sigma_x$, $\sigma_y$ e $\sigma_{xy}$ representam, respectivamente, a média de $x$, a média de $y$, a variância de $x$, a variância de $y$ e a covariância de $x$ e $y$. $C_1$ e $C_2$ são constantes. O valor da SSIM varia de 0 a 1, sendo 1 o melhor valor possível. Estudos sobre o desempenho da métrica SSIM têm apresentados bons resultados.

### A Métrica VQM

O algoritmo usado na métrica de qualidade visual (VQM - *Visual Quality Metric*) extrai características das imagens com medidas dos efeitos perceptuais de diversos distúrbios na sequência de vídeo, como exemplos, o ruído global e o borramento (YASAKETHU *et al.*, 2009). Essas medidas são combinadas em uma única métrica que fornece uma predição da qualidade global da imagem ou quadro de vídeo. O algoritmo VQM é dividido em quatro fases:

- Calibração – Correção de deslocamentos espaciais e temporais e desvios de contraste e brilho na sequência de vídeo processada.
- Extração de indicativos de qualidade – O conjunto de características de qualidade que caracteriza as mudanças de percepção nos domínios espacial, temporal, e da crominância são extraídos de sub-regiões espaço-temporais da sequência de vídeo.
- Estimação de parâmetros de qualidade – Nesta etapa, um conjunto de parâmetros de qualidade que descrevem as alterações perceptuais são calculados comparando características extraídas do vídeo processado com as características extraídas do vídeo de referência.
- Estimação da qualidade – A etapa final consiste em calcular uma métrica de qualidade geral usando uma combinação linear dos parâmetros calculados nas etapas anteriores.

### A Métrica PW-SSIM

O método PW-SSIM(*Perceptual Weighting Structural Similarity Index*) (REGIS *et al.*, 2015) utiliza a informação espacial perceptiva como forma de ponderação das regiões visualmente mais importantes. Esta ponderação é obtida do seguinte modo: primeiramente é calculada a magnitude dos vetores de gradiente no vídeo original, usando as máscaras de Sobel (FURNARI *et al.*, 2015), depois é gerado um quadro no qual os valores dos pixels so as magnitudes dos gradientes. Em seguida, esse quadro é particionado em blocos de $8 \times 8$ pixels e para cada bloco é calculado o SI,

$$\mathrm{SI} = \left( \frac{1}{N-1} \sum_{i=1}^{1} (\mu_S - S)^2 \right)^{\frac{1}{2}}, \tag{3.12}$$

em que, $\mu_S$ representa o valor médio da magnitude do gradiente em um bloco e $N$ é o número de *pixels* no bloco. Baseado nessa consideração, o valor do SI foi incorporado ao SSIM, levando ao modelo denominado Índice de Semelhança Estrutural com Ponderação Perceptual (*Structural Similarity Index with Perceptual Weighting*),

$$\text{PW-SSIM}(f,h) = \frac{\sum_{d=1}^{D} \mathrm{SSIM}_d(f,h) \cdot \mathrm{SI}_d}{\sum_{d=1}^{D} \mathrm{SI}_d}. \tag{3.13}$$

### A Métrica TPW-SSIM

Baseando-se nos resultados das simulações com a Equação 3.13, foi proposta em (REGIS, 2013) a métrica *Temporal Perceptual Weighted Video Quality Approach* (TPW-SSIM) que utiliza informações temporais de quadros vizinhos. No algoritmo proposto a qualidade temporal (TP-VQI) estimada por meio do índice de $PW-SSIM$ entre as diferenças dos quadros ($\Delta_{f,n}$ e $\Delta_{h,n}$), que são calculados como se segue:

$$\text{TP-VQI} = \frac{1}{N-1}\sum_{n=0}^{N-2}\text{PW-SSIM}(\Delta_{f,n},\Delta_{h,n}) \tag{3.14}$$

$$\Delta_{f,n} = |f_{n+1} - f_n|, \Delta_{h,n} = |h_{n+1} - h_n|. \tag{3.15}$$

Dessa forma, a TPW-SSIM é dada pela média entre o índice de qualidade espacial e o temporal,

$$\text{TPW-SSIM} = \textit{PW-SSIM} + \text{TP-VQI2}. \tag{3.16}$$

Para avaliação, implementação e depuração do código fonte do codificador HM 16.7 são utilizadas nesta tese as ferramentas Microsoft Visual Studio Express, Eclipse e Understand C++, que possuem capacidade de análise visual de fluxo do código, para facilitar e acelerar o entendimento do sistema.

## 3.8 A Estimação de Movimento Baseada em Blocos

Os algoritmos de busca de blocos são a técnica mais popular usada na estimação de movimento de codificadores de vídeo. Em geral, esses algoritmos se baseiam na divisão do quadro de luminância em macroblocos não sobrepostos de tamanho $N \times N$ que, por sua vez, são comparados com o macrobloco correspondente e seus vizinhos adjacentes no quadro de referência para criar um vetor que estipula a sua movimentação, encontrando o macrobloco correspondente do mesmo tamanho $N \times N$ na área de busca no quadro de referência. A posição do macrobloco correspondente no quadro de referência dá o vetor de movimento (MV) do macrobloco corrente. Este vetor de movimento é constituído das coordenadas $(x,y)$ do canto esquerdo-superior do macrobloco corrente representando as coordenadas iniciais do vetor e das coordenadas $(x,y)$ do canto esquerdo-superior do macrobloco do quadro de referência. Essas coordenadas podem ser positivas ou negativas. Um valor positivo significa um movimento para a direita ou um movimento descendente e um valor negativo significa um movimento para a esquerda ou movimento ascendente.

Esses vetores de movimento são usados no decodificador para predizer um novo quadro a partir do quadro de referência. Esse processo é chamado de compensação de movimento. A métrica usada é geralmente determinada usando uma das medidas de distorção de blocos

(BDM – *Block Distortion Measure*) como a diferença média absoluta (MAD – *Mean Absolute Difference*), a soma das diferenças absolutas (SAD – *Sum of Absolute Differences*) ou erro médio quadrático (MSE – *Mean Squared Error*). O macrobloco com o menor custo, resultante de uma dessas métricas, é considerado o correspondente ao macrobloco corrente.

Em geral as buscas são realizadas em blocos de tamanhos $16 \times 16$, que coincide com o tamanho dos macroblocos. Mas esse valor não é fixo, podendo ser configurados tamanhos menores ou maiores para o bloco de busca. Blocos de tamanho menores devem produzir melhores resultados de compensação de movimento. No entanto, um tamanho menor do bloco conduz a uma maior complexidade computacional (número de operações de busca que devem ser efetuadas) e um aumento no número de vetores de movimento que, em seguida, devem ser transmitidos. O envio de cada vetor de movimento acarreta o envio de mais *bits* e essa sobrecarga pode superar o benefício da energia residual reduzida. Um compromisso eficaz é adequar o tamanho dos blocos para as características da imagem, por exemplo, a escolha de regiões homogêneas de um quadro e escolher um tamanho de bloco pequeno em áreas de elevada riqueza de detalhes e movimento mais complexo (GROUP, 2008).

Por ter a capacidade de analisar em tempo real os valores dos *pixels* e os parâmetros de configuração do codificador, o ABMA funciona como um aliado de outros processos de análise e algoritmos de estimação de movimento. A eficiência desse esquema de escolha adaptativa é diretamente correlacionada com a métrica escolhida. Para testes, a escolha do algoritmo é feita usando o valor da PSNR do quadro anterior. O algoritmo escolhido tende a ser mais robusto e preciso em função do aumento da degradação do vídeo decodificado.

Essa capacidade de análise em tempo real permite que a escolha do algoritmo seja adequada para cada momento da sequência de vídeo. Considerando um vídeo 3D, formado pela imagem monoscópica e sua informação de profundidade, que é representada por uma imagem em níveis de cinza. O esquema ABMA oferece diferentes algoritmos de estimação de movimento para a imagem monoscópica e a imagem de informação de profundidade.

## CAPÍTULO 4

# Aprendizagem Automática

Aprendizagem automática ou aprendizado de máquina tem recebido um grande enfoque de várias de pesquisas nos últimos anos, principalmente nas áreas da inteligência artificial, reconhecimento de padrões e processamento de sinais, devido à capacidade de aprender a partir de grandes volumes de informação e propor melhores soluções para diversos problemas.

O aumento da capacidade computacional impulsionou e disseminou o uso de técnicas de aprendizado automática para diversas aplicações. Considerando que, atualmente, é comum encontrar dispositivos com, por exemplo, mais de oito núcleos de processamento e memória volátil em abundância, tornou-se viável levar essas técnicas para aplicações nas mãos de usuários finais.

A Aprendizagem automática é uma área da Inteligência Artificial cujo objetivo é o desenvolvimento de técnicas computacionais sobre o aprendizado, bem como a construção de sistemas capazes de adquirir conhecimento de forma automática. Esses sistemas de aprendizado são programas de computador que tomam decisões baseados em experiências acumuladas por meio da solução bem sucedida de problemas anteriores e por meio da classificação de um conjunto de amostras de variáveis do sistema em análise. Essas técnicas são ferramentas poderosas para aquisição automática de conhecimento, entretanto, não existe um único algoritmo que apresente melhor desempenho para todos problemas.

Os algoritmos de aprendizagem automática procuram padrões de um conjunto de dados. Esses algoritmos existem há bastante tempo, no entanto, a infraestrutura de tecnologia de informação implantada nos dias de hoje aumenta significativamente a quantidade de dados digitais disponíveis para alimentar esses algoritmos. Isso acontece devido a dois fatores: a informatização em massa e o surgimento da Internet.

O problema de modelagem de dados é pertinente a muitas aplicações da Engenharia. Em modelagem de dados empíricos, é usado um processo de indução para construir um modelo matemático capaz de expressar as relações entre entrada e saída, a partir das quais são deduzidas respostas ainda não observadas no conjunto de treinamento. Basicamente, a quantidade e qualidade dos dados disponíveis governam o desempenho deste modelo empírico. Por corresponder

a uma técnica de aprendizado baseada em amostragem de dados, o conjunto de dados fornece tratabilidade computacional, mas conduz à uma amostragem esparsa do espaço de entrada.

Conhecer o tipo e as características dos dados que serão objetos de estudo também é fundamental para a escolha dos métodos mais adequados. Pode-se categorizar os dados em dois tipos: quantitativos e qualitativos. Os dados quantitativos são representados por valores numéricos. Esses dados podem ser discretos e contínuos. Já os dados qualitativos contêm os valores nominais e ordinais (categóricos). Em geral, antes de se aplicar os algoritmos de mineração é necessário explorar, conhecer e preparar os dados.

Um problema de aprendizagem deve ter uma medida que mede a eficiência de um algoritmo ao executar uma tarefa de otimização. Um objetivo universal para qualquer algoritmo de aprendizado supervisionado é a generalização, que estima a precisão que serão previstos dados futuros. A função que calcula o erro de generalização é representada por $g(\theta)$, em que $\theta$ é o vetor de parâmetros. Devido a dificuldade de se obter o cálculo de $g(\theta)$, ele é substituido pela aproximação $\hat{g}(\theta)$ calculada no conjunto de dados de treinamento $\{(x_i, y_i)\}_{i=1}^{n}$, chamada de erro de treinamento e calculado pela Equação

$$\hat{g}(\theta) = \frac{1}{n}\sum_{i=1}^{n} l(x_i, y_i; \theta). \tag{4.1}$$

em que $x_i$ representa o vetor de atributos e $y_i$ a informação de classificação do vetor de atributos.

Ao tentar minimizar a perda sobre o conjunto de treinamento existe o perigo do sobreajuste. Uma maneira comum para prevenir o sobreajuste durante o treinamento é a técnica de regularização. Regularização refere-se a aumentar a função objetivo $\hat{g}(\theta)$ com um termo extra que penaliza vetores $\theta$ complexos. Esse termo é incorporado na Equação 4.1 reprentando-a como a soma entre a minimização do erro e o termo de regularização, resultando em

$$\hat{g}(\theta) = \frac{1}{n}\sum_{i=1}^{n} l(x_i, y_i; \theta) + r(\theta). \tag{4.2}$$

## 4.1 Terminologia

Nesta seção será apresentada a terminologia da área de aprendizagem automática que será utilizada no texto desta tese.

### 4.1.1 Exemplo

Exemplo, padrão ou instância caracterizam um objeto único a partir do qual um modelo sera aprendido, ou sobre o qual um modelo será usado (por exemplo, para predição). Na maioria dos trabalhos sobre aprendizado de máquina, exemplos são descritos por vetores de características. Um exemplo padrão poderia ser uma amostra de tecido de um paciente, que poderia estar associada a presença de câncer.

### 4.1.2 Atributos

Cada exemplo, ou instância, que se comporta com uma entrada para um processo de aprendizagem de máquina é caracterizada por seus valores de um conjunto fixo e pré-definido de características ou atributos. O uso de um conjunto fixo de recursos impõe outra restrição sobre os tipos de problemas geralmente considerados em aprendizado de máquinas e mineração de dados. Considerando diferentes instâncias com características diferentes, por exemplo, veículos de transporte. Número de rodas é uma característica que se aplica a muitos veículos, mas não para os navios, por exemplo, dessa forma número de mastros pode ser uma característica que se aplica a navios, mas não para veículos terrestres. A solução padrão é caracterizar cada recurso possível como um atributo e usar um valor irrelevante para indicar que um determinado atributo não está disponível para um caso particular. O valor de um atributo de uma instância específica é uma medida da quantidade para o qual o atributo se refere. Há uma ampla distinção entre quantidades numéricas e aquelas que são nominais. Atributos numéricos, às vezes chamados atributos contínuos, são números representando uma medida real. Note-se que o termo contínuo é abusada rotineiramente neste contexto; atributos de valores inteiros, certamente, não são contínuos no sentido matemático. Atributos nominais podem assumir valores em um conjunto pré-determinado e finito de possibilidades e às vezes são chamados categóricos ou classificatórios.

## 4.2 Aprendizado Supervisionado

O aprendizado supervisionado é bastante comum em problemas de classificação, porque o objetivo é muitas vezes fazer o computador aprender um sistema de classificação previamente criado. Um exemplo comum de aprendizado supervisionado são sistemas de reconhecimento de dígitos. De forma geral, a aprendizagem supervisionada é adequada para qualquer problema em que a dedução da classificação pode otimizar o problema. Em alguns casos, pode até não ser necessário dar classificações pré-determinadas para cada instância de um problema se o agente pode entender as classificações por si.

Aprendizagem supervisionada é a técnica mais comum para o treinamento de sistemas de redes neurais e árvores de decisão. Ambas as técnicas são altamente dependente da informação dada pelas classificações pré-determinadas. No caso de redes neurais, a classificação é utilizada para determinar o erro da rede e, em seguida, ajustar a rede para a minimização do erro. Em árvores de decisões, as classificações são usadas para determinar os atributos que fornecem a maior parte da informação que pode ser utilizada para resolver problemas de classificação.

A Aprendizagem Supervisionada deduz uma função a partir de dados de treinamento. Os dados de treinamento consistem em um conjunto de objetos de entrada (normalmente vetores) e as saídas desejadas. A saída da função pode ser um valor contínuo, o que classifica um regressão, ou pode prever uma classe para cada objeto de entrada, o que caracteriza uma classi-

ficação. A tarefa do algoritmo supervisionado é prever o valor da função para qualquer objeto de entrada válido depois de ter visto uma série de exemplos de treinamento. Para conseguir isso, o algoritmo precisa generalizar a partir dos dados apresentados a situações não conhecidas de uma forma racional.

## 4.3 Aprendizado Não-Supervisionado

Aprendizado não supervisionado é realizado quando, para cada exemplo, apenas os atributos de entrada estão disponíveis. Essas técnicas de aprendizado são utilizadas quando o objetivo for encontrar em um conjunto de dados padrões, tendências (aglomerados) que auxiliem o entendimento desses dados. Uma das tarefas mais comuns de algoritmos não-supervisionados é a de classificação de entradas. O propósito de um algoritmo de classificação não-supervisionado é identificar a qual classe um determinado registro pertence. Por exemplo, categoriza-se cada registro de um conjunto de dados contendo as informações sobre os colaboradores de uma empresa: Perfil Técnico, Perfil Negocial e Perfil Gerencial. O modelo analisa os registros e então é capaz de dizer em qual categoria um novo colaborador se encaixa. A tarefa de classificação pode ser usada por exemplo para:

- Determinar quando uma transação de cartão de crédito pode ser uma fraude;
- Identificar em uma escola, qual a turma mais indicada para um determinado aluno;
- Diagnosticar onde uma determinada doença pode estar presente;
- Identificar quando uma pessoa pode ser uma ameaça para a segurança.

## 4.4 Aprendizagem em Larga Escala

O aprendizado em larga escala necessita de muitos recursos computacionais para operar seus algoritmos sobre grandes quantidades de dados. O aumento da capacidade computacional, devido ao avanço tecnológico e à queda dos preços dos computadores, facilita o uso de técnicas de aprendizagem em larga escala atualmente para várias aplicações.

Um ponto de interesse do problema analisado nesse estudo é o tamanho do conjunto de treinamento. De forma geral, se o conjunto de treinamento em análise não puder ser armazenado em um sistema computacional de uma determinada aplicação, ele pode ser classificado como um conjunto de treinamentos de dados em larga escala. Outra definição apropriada para aprendizado em larga escala seria que consistm em analisar problemas nos quais o pricinpal limitante computacional é o tempo disponível. Um grande conjunto de treinamento representa um desafio para a complexidade computacional de um algoritmo de aprendizagem automática. Algoritmos viáveis em tais conjuntos de dados devem escalar, na pior das hipóteses, linearmente com o crescimento do número de exemplos.


**Figura 4.1** Relação gráfica entre as principais resoluções usadas em sistemas de TVD.

De certa forma, os sistemas de codificação de vídeos desenvolvidos até hoje apresentam carga de processamento e consumo de memória compatíveis com as plataformas de hardware atuais. Entretanto, o tempo de processamento da compressão de vídeos em UHD (3840×2160 e 7680×4320) tem sido um gargalo para diversas aplicações de vídeo em tempo real, chegando a inviabilizar algumas aplicações de transmissões em tempo real ou codificação de vídeo com alta qualidade em dispositivos móveis. A Figura 4.1 ilustra a relação entre o número de linhas vídeo mais usados, em resoluções com relação de aspecto 16:9, e pode-se entender o aumento da quantidade de dados entre as resoluções. Para a maioria dos codificadores, a transmissão em tempo real só é viável quando os parâmetros de qualidade do processo de codificação são reduzidos para que o tempo de codificação seja compatível com a demanda e a capacidade do canal de transmissão da aplicação. Dessa forma, a redução do tempo de codificação, ou complexidade computacional do processo, passa a ser um problema significativo para sistemas de transmissão de vídeo.

Existem,, ao menos, três abordagens para resolver um problema de aprendizagem em larga escala (LANGFORD *et al.*, 2009):

- Tratar os dados de treinamento como um fluxo de exemplos e aplicar um algoritmo de aprendizagem em cada exemplo de acordo com a ordem cronológica. O modelo é incrementado a cada novo ciclo. Tais algoritmos têm requisitos de espaço de armazenamento desprezíveis ou levemente dependente do número de exemplos e da quantidade de atributos e não precisam armazenar todo o conjunto de treinamento em memória. O sistema proposto nessa tese utiliza essa abordagem.

- Paralelizar um algoritmo de aprendizagem em lote de dados, o que permite dividir um grande problema de aprendizagem em uma série de problemas menores. Dado um nú-

mero de máquinas, ou núcleos de processadores, suficientes, cada sub-problema se torna de menor complexidade, portanto, tratável.

- Pode-se pré-processar os dados de treinamento e retirar um pequeno subconjunto de dados para treinar. Este subconjunto pode ser escolhido de forma aleatória, ou usando uma abordagem mais sofisticada que tenta coletar apenas exemplos informativos. Dessa forma, pode-se reduzir o conjunto de treinamento suficientemente para tornar a algoritmo de treinamento tratável.

## 4.5 O Método do Gradiente Descendente

O método do gradiente descendente é uma técnica matemática clássica para encontrar o mínimo de uma função $f(\theta)$. Esse método usa o fato de que o gradiente $\nabla f$ de uma função aponta para a direção do maior crescimento. Isso significa que $-\nabla f$ aponta na direção oposta e, então, um algoritmo para para achar o mínimo de $f$ pode ser definido atualizando uma estimativa de $\theta_t$ usando

$$\theta_{t+1} = \theta_t - \eta_t \nabla f(\theta_t), \tag{4.3}$$

em que $\eta_t$ é a taxa de aprendizagem, que determinada a velocidade que o resultado se move em direção ao gradiente. É importante escolher esse valor cautelosamente, porque se for muito pequeno, a convergência é lenta, e se for muito grande, então pode-se não encontrar o mínimo da função. Para o problema do aprendizado automático, tem-se que

$$\nabla \hat{g}(\theta_t) = \frac{1}{n} \sum_{i=1}^{n} \nabla l(x_i, y_i; \theta_t) + \nabla r(\theta_t), \tag{4.4}$$

## 4.6 Gradiente Estocástico

A complexidade computacional dos algoritmos de aprendizagem automática torna-se o fator limitante crítico quando se prevê a utilização de grandes conjuntos de dados. Esta condição favorece algoritmos de gradiente estocásticos para problemas de aprendizado de máquina em grande escala. No método do gradiente descendente explicado na seção 4.5, o cálculo é feito utilizando todos exemplos do conjunto de treinamento. Uma alteração simples deste é encontrar o gradiente com respeito a um único exemplo escolhido aleatoriamente. Essa técnica é chamada de gradiente estocástico descendente (SGD - *Stochastic Gradient Descent* e a Equação de atualização do vetor de parâmetros $\theta_t$ é

$$\theta_{t+1} = \theta_t - \eta_t \nabla l(x_{i(t)}, y_{i(t)}; \theta_t) - \eta_t \cdot \nabla r(\theta_t), \tag{4.5}$$

Dessa forma apenas uma aproximação do gradiente verdadeiro é obtida ao usar apenas um exemplo do cojunto de dados de treinamento. Portanto, não é mais garantido que o gra-

diente se mova na direção da minimização em toda iteração. No entanto, há pelo menos duas razões importantes pelas quais o método do gradiente estocástico se torna útil em problemas de apredizagem, que são:

- é significantemente mais rápido do que o gradiente descendente tradicional quando o número de exemplos no conjunto de treinamentos é grande e
- pode-se ser demonstrado que os métodos de gradiente estocástico descendente minimizam o erro de generalização mais rápido que o gradiente descendente.

Uma das principais caractéristicas dos algoritmos de aprendizagem automática baseados em gradiente estocástico, e que é usada nessa pequisa, é que eles não requerem o conhecimento *a priori* das estatísticas das amostras do sinal de vídeo, pois fazem uso de estimativas instantâneas do gradiente (obtidas a partir de dados instantâneos). Logo, pode-se dizer que esses algoritmos incorporam mecanismos de aprendizagem que permitem rastrear variações nas características dos sinais de vídeo digital.

## 4.7 Máquinas de Vetores Suporte

As máquinas de vetores suporte ou SVM (*Support Vector Machine*), são algoritmos de aprendizagem bastante utilizados na área de aprendizagem de máquina. SVM pode ser visto como uma extensão do algoritmo Perceptron (SHALEV-SHWARTZ *et al.*, 2007), pois, da mesma forma, tenta encontrar um hiperplano que separe os dados, classificando-os eficientemente.

O algoritmo Perceptron simplesmente tenta encontrar qualquer hiperplano que separe os dados, sem considerar como o hiperplano separa os dados. Mas intuitivamente, um hiperplano que é tão longe quanto possível de qualquer classe é preferível, porque espera-se que este hiperplano possa generalizar melhor os dados fora do conjunto de treinamento. A medida de como um hiperplano separa os dados é a sua margem. Esta é a distância do hiperplano ao ponto mais próximo no conjunto de dados. Uma margem grande significa que o hiperplano separa eficientemente os dados, reduzindo a probabilidade de erro de classificação. A Figura 4.2 ilustra a separação dos dados e as margens.

Os vetores suporte são os pontos de treinamento que não estão classificadas com confiança, isto é, eles são classificados erroneamente ou quando classificados corretamente, pertecem a uma região de margem de erro de classificação. De acordo com a Equação 4.6, o vetor peso ideal $w$ é uma combinação linear de vetores suporte. Portanto, os vetores suporte são os pontos de treinamento que minimizam o erro de classificação e, por sua vez, o objetivo da máquina de vetores suporte é descobri-los. Dado um conjunto de treinamento $\{(x_i, y_i)\}_{i=1}^{n}$, em que $y \in \{\pm 1\}$, o hiperplano parametrizado pelo vetor normal $w$ que equilibra a meta de separação de dados e maximiza as margens pode ser encontrado pela solução da problema de otimização

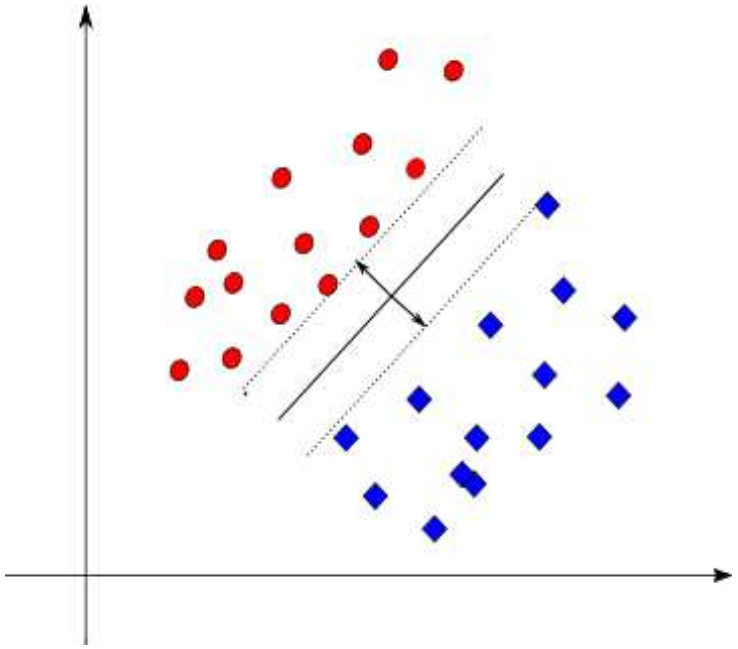


**Figura 4.2** Hiperplano e margens encontrados pelo algoritmo SVM.

$$\min_{w} \frac{\lambda}{2} \|w\|^2 + \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y(w \cdot x)), \tag{4.6}$$

em que $\lambda \geq 0$ é o parâmetro de regularização e é definido por

$$\lambda = \frac{1}{nC}. \tag{4.7}$$

É possível utilizar diferentes tipos de funções *kernel*, que é uma função similaridade, descritos na Subseção 4.7.1, e diferentes parâmetros a variar de acordo com o *kernel* escolhido. Além disso, um melhor parâmetro para uma base, não necessariamente e muito provavelmente não será o melhor parâmetro para outra. Dessa forma identifica-se uma área de estudo na busca dos melhores parâmetros para uso do algoritmo SVM.

As princpais vantagens do uso de máquinas de vetores suporte são (MENON, 2009):

- Generalização: Os resultados da generalização do modelo são, na maioria dos resultados, satisfatórios. A capacidade de generalização de um classificador é medida por sua eficiência na classificação de dados que não pertençam ao conjunto utilizado em seu treinamento. Dessa forma, o modelo resultante de um algoritmo SVM tende a apresentar um reduzido sobreajuste.

- Robusto para dados de grandes dimensões: O algoritmo SVM tem se mostrado robusto para dados de grandes dimensões como vetores de *pixels* de imagens. A boa capacidade de generalização é um ponto fundamental para esse tipo de aplicação.

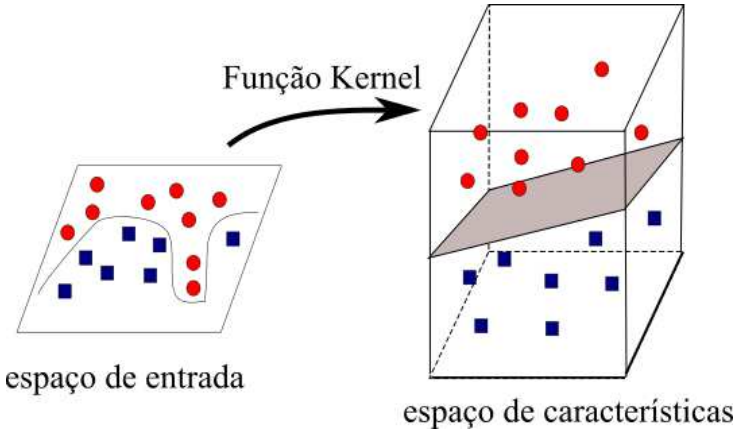



**Figura 4.3** Transformação de um problema não-linearmente separável em um problema linearmente separável usando uma função kernel.

- Convexidade: A aplicação das SVMs implica na otimização de uma função quadrática, que possui apenas um mínimo global, diferentemente das Redes Neurais Artificiais em que há a presença de mais de um mínimo local.
- Teoria consolidada: A base teórica desse método é bem fundamentada e consolidada.

### 4.7.1 As Funções *Kernel*

Matematicamente, uma função *kernel* $K$ é representada, em um espaço bidimensional, por uma equação que recebe dois pontos $x_i$ e $x_j$ e calcula o produto escalar $\Phi(x_i)$ e $\Phi(x_j)$ no espaço de características, em que $\Phi_n$, com $n = \{0,...,m\}$ e $m$ sendo o número de atributos, são funções que mapeiam os dados do espaço de entrada para um novo espaço, chamado espaço de características. Sendo assim, uma função *kernel* é definida em

$$K(x_i,x_j) = \Phi(x_i) \cdot \Phi(x_j) \tag{4.8}$$

De forma geral, a função *kernel* é mais simples que a função $\Phi$, pois na maioria dos casos, $\Phi$ assume formas complexas, em função do número de atributos. Por este motivo, é comum definir a função *kernel* sem conhecer explicitamente o mapeamento de $\Phi$. A utilidade das funções *kernel* está, portanto, na simplicidade de cálculo e na capacidade de representar espaços muito abstratos. A Figura 4.3 ilustra o conceito de separação dos dados no espaço de caracterísiticas. Resumidamente, as funções *kernel* mais utilizadas são listadas na Tabela 4.1. O algoritmo implementado nesta tese usa o *kernel* gaussiano.

#### O *Kernel* Função de Base Radial

O *kernel* RBF (*Radial Basis Function*) é bastante utilizado para resolução de problemas de aprendizagem, inclusive é usado computacionalmente como padrão em muitas bibliotecas de linguagens de programação que utilizam o algoritmo SVM e foi o tipo de *kernel* utilizado

**Tabela 4.1** Tipos de funções *kernel* mais utilizadas.

| **Tipo de *kernel*** | **Função *kernel*** |
|---|---|
| Polinomial | $({x_i}^T \cdot x_j + 1)^p$ |
| Gaussiano | $\exp\left(-\frac{1}{2\sigma^2}\left\|x_i - x_j\right\|^2\right)$ |
| Sigmoidal | $\tanh(\beta_0 x_i \cdot x_j + \beta_1)$ |

nesta tese. Na máquina RBF, diferentemente do *kernel* linear, é possível resolver problemas, originalmente, não linearmente separáveis, através do mapeamento para um espaço de maior dimensão. Na ausência de conhecimento prévio, o *kernel* RBF é uma das melhores opções quando a solução não é linear.

Existem dois parâmetros que podem ser variados em busca de um melhor resultado para o aprendizado do classificador, são eles: $\gamma$ e $C$ (CHEN *et al.*, 2014). $\gamma$ é o parâmetro livre do *Kernel* gaussiano e $C$ é a variável de controle do custo de classificação. Intuitivamente, o parâmetro $\gamma$ define quão longe a influência de um único exemplo de treinamento atinge. Valores baixos determinam uma influência pequena e valores elevados, o contrário. O parâmetro $\gamma$ pode ser visto como o inverso do raio de influência de amostras selecionadas pelo modelo como vetores de suporte. O valor do parâmetro $C$ balanceia a má classificação dos exemplos de treinamento em relação à simplicidade da superfície da decisão. Um valor baixo de $C$ faz com que a superfície de decisão seja mais suave, enquanto que altos valores de $C$ leva o modelo a tentar classificar todos os exemplos de treinamento corretamente, dando ao modelo liberdade para escolher mais amostras de vetores de suporte (HSIEH *et al.*, 2015), mas aumentando os efeitos de sobreajuste.

## 4.8 Aprendizagem Automática em Tempo Real

Métodos de aprendizagem automática em tempo real, também conhecida como de aprendizagem *on-line*, estão relacionados com métodos estocásticos por operarem em apenas um único exemplo em cada iteração. Além disso, muitas regras de aprendizagem em tempo real, incluindo a regra Perceptron, podem ser vistas como a implementação de um gradiente estocástico. Muitos desses métodos, incluindo o Perceptron, também tem fortes ligações com a margem ou norma do preditor, embora não procurem diretamente minimizar o objetivo SVM. No entanto, algoritmos de aprendizado em tempo real foram propostos como alternativas rápidas para SVMs. Tais algoritmos podem ser usados para obter um preditor com baixo erro de generalização utilizando uma abordagem de processar exemplo por exemplo ou pequenos lotes de exemplos. No entanto, os esquemas de conversão de algoritmos de aprendizagem em tempo real não necessariamente produzem soluções precisas para o problema original e seu desempenho é normalmente inferior a otimizadores tradicionais, que usam todo o conjunto de dados para o treinamento. No entanto, com o crescimento do número de exemplos o desempenho de um sistema de treinamento em tempo real tende a se aproximar dos sistemas de treinamentos ex-

ternos, com a significativa vantagem de possuir maior capacidade de adaptação à mudança das características dos dados ao longo do tempo. É importante mencionar que o algoritmo Pegasos compartilha a simplicidade e rapidez de algoritmos de aprendizagem em tempo real (SHALEV-SHWARTZ *et al.*, 2007).

## 4.9 O Algoritmo Pegasos

Pegasos (*Primal Estimated sub-GrAdient SOlver for SVM*) é um algoritmo de otimização de máquinas de vetores suporte cujo tempo de execução é independente do número de exemplos no conjunto de treinamento (SHALEV-SHWARTZ *et al.*, 2007). O algoritmo opera em mini-lotes dos dados de treinamento disponíveis. O tamanho dos lotes, representado pela variável $k$, é determinado pelo usuário. Quando $k = 1$, Pegasos se comporta como um SGD e quando $k = n$, $n$ sendo o número de exemplos de treinamento, ele se comporta como uma projeção de subgradiente. No entanto, apesar de sua semelhança com SGD tradicionais, uma importante propriedade do algoritmo Pegasos é que ele converge para a solução aproximada com precisão $\rho$ em $\tilde{O}(\frac{d}{\lambda\rho})$ iterações, em que $d$ é o número de atributos diferentes de zero em cada exemplo, enquanto que os métodos baseados em SGDs tradicionais convergem em $\tilde{O}(\frac{1}{\rho^2})$.

O algoritmo procede da seguinte forma: Primeiramente o vetor $w_1$ é inicializado com o valor 0. Em cada iteração $t$, um exemplo é extraído do conjunto de treinamento $(x_{i_t}, y_{i_t})$ através da escolha aleatória do índice $i_t \in \{1, ..., m\}$. Subsituindo em 4.10, tem-se:

$$f(w; i_t) = \frac{\lambda}{2} \|w\|^2 + l(w; (x_{i_t}, y_{i_t})), \tag{4.9}$$

tendo seu gradiente calculado por

$$\nabla_t = \lambda w_t - \mathbb{1}\left[y_{i_t} \langle w_t, x_{i_t} \rangle < 1\right] y_{i_t} x_{i_t}. \tag{4.10}$$

em que $\mathbb{1}\left[y \langle w, x \rangle < 1\right]$ é a função indicador que retorna o valor um se o argumento é verdadeiro e zero em caso contrário. Depois disso atualiza-se $w_{t+1} \leftarrow w_t - \eta_t \nabla_t$, usando $\eta_t = \frac{1}{\lambda t}$.

## 4.10 Seleção Automática de Atributos

Nos últimos trinta anos, a dimensionalidade dos dados envolvidos em tarefas de aprendizagem automática e mineração de dados tem aumentado significantemente. Dados com dimensionalidade muito elevada apresentam sérios desafios aos métodos existentes de aprendizagem, levando à um problema conhecido como a maldição da dimensionalidade (KOUIROUKIDIS; EVANGELIDIS, 2011). Com a existência de um grande número de atributos, um modelo de aprendizagem tende ao superajuste e seu desempenho de aprendizagem degenera. Para resolver o problema da dimensionalidade, técnicas de redução de dimensionalidade têm sido estudadas,

que formam um ramo importante na área de pesquisa de aprendizado de máquina. A seleção automática de atributos é uma das técnicas mais utilizadas em aplicações profissionais para reduzir a dimensionalidade dos dados. Essa técnica destina-se a escolher um pequeno subconjunto dos atributos de acordo com um determinado critério de avaliação de relevância, o que geralmente leva a um melhor desempenho de aprendizagem, menor custo computacional, e melhor modelo de interpretabilidade dos dados. As técnicas de seleção de atributos tem sido aplicadas com sucesso em muitas aplicações reais, como reconhecimento de padrões, a categorização de texto, processamento de imagem e bioinformática.

O problema fundamental da aprendizagem automática é aproximar o relacionamento funcional $f()$ entre uma entrada $X = x_1, x_2, ..., x_M$ e uma saída $Y$, baseado na memória dos pontos de dados. Algumas vezes, $Y$ pode não precisar do conjunto completo dos $M$ atributos definidos em $X$ e pode ser determinado por um subconjunto $x_1, x_2, ..., x_m$ onde $m \leq M$. Na prática, existem dois problemas acarretados pela presença de atributos irrelevantes no processo de apredizagem:

- Os atributos irrelevantes induzem um aumento no custo computacional. De forma geral, com o aumento de atributos o custo computacional tende a aumentar polinomialmente.
- A presença de atributos irrelevantes no processo de apredizagem automática tendem a um sobreajuste do modelo.

### 4.10.1 A Técnica de Seleção de Atributos – IGAE

Diferentes técnicas de classificação e seleção de atributos foram propostas na literatura de aprendizagem automática. O objetivo destas técnicas é descartar características irrelevantes e redundantes a partir de um dado vetor de características. Nesta tese, será considerada a técnica IGAE (CORREA *et al.*, 2015) (*Information Gain Attribute Evaluation*).

A entropia é um medida comumente usada na teoria da informação, que caracteriza a organização de uma coleção arbitrária de exemplos. A maioria das técnicas que medem o ganho de informação utilizam essa grandeza como forma de avaliação. A medida de entropia é considerado como uma medida da imprevisibilidade do sistema. A Entropia de $Y$ é dada por (SHANNON, 1948)

$$H(Y) = -\sum_{y \in Y} (p(y)) \log_2 (p(y)) \tag{4.11}$$

em que $p(y)$ é a função densidade de probabilidade de $Y$. Se os valores observados de $Y$ no conjunto de dados são divididos de acordo com os valores de um segundo atributo $X$, e a entropia de $Y$ em relação às partições induzidas por $X$ é menor do que a entropia de $Y$ antes de

particionamento, então há uma relação entre as características $Y$ e $X$. Dessa forma, a entropia de $Y$ depois de $X$ ser observado é dada por (SHANNON, 1948)

$$H(Y|X) = -\sum_{x \in X}(p(y))\sum_{y \in Y}(p(y/x))\log_2(p(y/x)) \tag{4.12}$$

em que $p(y|x)$ é a probabilidade condicional de $y$ dado $x$. Com isso, pode-se definir uma medida que reflete informações adicionais sobre Y fornecido pelos valores de X que representam o montante pelo qual a entropia de Y diminui. Dessa forma $IG$ é definido como

$$IG = H(Y) - H(Y|X) = H(X) - H(X|Y). \tag{4.13}$$

A fraqueza do critério $IG$ é que ele é tendencioso em favor de vetores de atributos maiores, ou seja com mais valores, mesmo quando eles não são mais informativos.

# CAPÍTULO 5

# Metodologia

## 5.1 O Documento JCTVC-L1100

O comitê JCT-VC definiu, em um das suas reuniões de padronização do HEVC, o documento JCTVC-L1100 (BOSSEN, February, 2012), que descreve os procedimentos para garantir condições comuns de testes para propostas de novas funcionalidades. As condições de teste comuns incluem um conjunto de sequências de vídeo de teste com várias resoluções, diferentes conteúdos, e várias configurações de codificação predefinidas para aplicações típicas. Na fase de desenvolvimento do HEVC, foram utilizados três tipos de configurações de codificação que estão definidas no documento JCTVC-L1100 (CTC), para serem utilizadas para experimentos fundamentais entre as reuniões. As configurações são:

- *All Intra (AI)*: Todas os quadros são codificados de forma independente com apenas predição intra-quadro. Esta configuração é utilizada para aplicações como edição de vídeos onde é necessário o acesso a todos os quadros da sequência.

- *Random Access (RA)*: Codificação com elevadas taxas de compressão compressão. A ordem de codificação e a ordem de saída dos quadros diferem, induzindo um atraso de codificação. Esta configuração representa aplicações de radiodifusão e de *streaming*.

- *Low Delay P and B (LP and LB)*: Codificação sem o atraso de codificação estrutural apresentado quando usado os parâmetros RA. Esta configuração é avaliada usando apenas quadros P (predição uni-direcional) ou apenas quadros B (predição uni-direcional e bi-direcional).

Essas três definições de configurações de testes são avaliadas em uma configuração correspondente ao perfil principal do HEVC (*Main Profile*) parametrizada para a mais alta eficiência de codificação. Para a avaliação do teste, sequências de vídeo com duração de 10 segundos são comumente usadas. Segundo as especificações no documento JCTVC-L1100, esta duração é considerada suficiente para uma avaliação qualificada da qualidade visual da sequência

reconstruída. Neste trabalho, em função da reduzida capacidade de processamento disponível para as simulações, foi realizada uma avaliação para entender quantos quadros seriam necessários para verificar o patamar mínimo do erro de classificação do modelo gerado. Considerando o conjunto de vídeos utilizados, observou-se que 130 quadros é um número suficiente para esse fim.

Para a avaliação, todas as configurações de teste são aplicadas a todos as sequências disponíveis com quatro valores diferentes do parâmetro de quantização (QP – *Quantization Parameter*) 22, 27, 32 e 37.

Deve-se notar que as aplicações reais normalmente aplicam estratégias de controle de taxa e de otimização de acordo com as necessidades de cada aplicação. O controle do codificador foi concebido para satisfazer as necessidades da aplicação, em vez de responder às condições de codificação experimentais, que em geral, são bem controladas. As configurações de teste predefinidas são uma ferramenta para a avaliação da atividade de padronização, a fim de proporcionar formas de comparação justas entre diferentes propostas. Ao avaliar esquemas de codificação com base em diferentes implementações de *software*, muitas vezes não é fácil concluir sobre o desempenho de qualquer um dos sistemas concorrentes dado que normalmente as implementações utilizam estratégias de otimização diferentes. A padronização das configurações proporciona comparações eficientes.

## 5.2 A Biblioteca Dlib-ml

O algoritmo SVM, apesar de ser baseado em um conceito intuitivamente simples, possui implementação complexa. Uma possível implementação desse algoritmo demandaria um número elevado de horas de desenvolvimento e, principalmente, otimização do tempo de execução. Existem algumas opções disponíveis para *download* gratuito. Entre elas, se destaca a biblioteca Dlib-ml por sua facilidade de uso e desempenho. Por esses motivos ele foi escolhida para ser integrada ao código nativo da implementação de referência HM.

O algoritmo Pegasos A Dlib-ml é uma biblioteca de código aberto implementada na linguagem de programação C++ para se adequar, sem grandes dificuldades, a diferentes plataformas e arquiteturas de *hardware*. Seu *design* é fortemente influenciado pelas idéias de projeto por contrato e engenharia de software baseada em componentes. Isso significa que ela é antes de tudo uma coleção de componentes de *software* independentes, cada um acompanhado por uma extensa documentação e modos de depuração distintos. Além disso, a biblioteca tem a intenção de ser útil em projetos de pesquisa e comerciais, sendo cuidadosamente projetada para tornar mais fácil a integração em aplicações C++.

Há uma boa quantidade de bibliotecas de aprendizagem automática difundidas na Internet. No entanto, muitas dessas bibliotecas concentram-se em fornecer um bom ambiente desenvolvimento usando outras linguagens de programação diferentes de C++. Dois exemplos deste tipo de projeto são os conjuntos de ferramentas densenvolvidos em Shogun e Tocha que,

mesmo sendo implementados em C ++, não estão focados em fornecer apoio para o desenvolvimento de *software* de aprendizagem de máquina nessa linguagem. Em vez disso, esses conjuntos são destinados principalmente para serem usados com linguagens como R, Python, Matlab, ou Lua. Dessa forma, existem conjuntos de ferramentas, tais como a Dlib-ml que são explicitamente dirigidas a usuários que desejam desenvolver *software* em C++. Dadas essas considerações, a biblioteca Dlib-ml procura para ajudar a preencher algumas das lacunas em apoio ferramental ao desenvolvimento.

## 5.3 Indicadores de Desempenho

A análise de desempenho do processo de codificação HEVC é, em geral, uma tarefa complexa, uma vez que pode ser realizada de várias maneiras diferentes com base em, por exemplo, eficiência de compressão, complexidade, qualidade visual, aplicação da relação taxa/distorção (RDO), atrasos, robustez, etc. O objetivo deste Capítulo é apresentar a eficiência de compressão da implementação proposta $\Omega$ em comparação com o a implementação de referência do codificador HEVC, em sua versão 16.7, tanto em termos de avaliações objetivas de qualidade e da complexidade computacional. Idealmente seria importante incluir uma etapa de avaliação subjetiva, dado que baseando-se unicamente em avaliações objetivas de qualidade pode-se, em alguns casos, subestimar a redução na taxa de bits e, portanto, afetar a análise da eficiência de compressão. No entanto, devido ao dispendioso processo necessário para avaliações subjetivas, o escopo desse trabalho se restringirá a usar métricas objetivas de avaliação de qualidade, como a PSNR, a BD-Rate e a BD-PSNR.

Os indicadores de desempenho utilizados nesse trabalho serão a eficiência de compressão e a complexidade do codificador. A eficiência de compressão é medida através da métrica *Bjøtegaard Delta Rate* (BD-Rate), calculada conforme o processo descrito em (BJONTEGAARD, 2001) entre o codificador de referência e a implementação proposta nessa tese. Nesse contexto, valores negativos da métrica BD-Rate correspondem a ganhos de eficiência de compressão e, considerando o uso do formato de subamostragem de crominância 4:2:0, a BD-Rate será calculada apenas para a componente de luminância. A duração do processamento de codificação de um vídeo, em segundos, é usado como métrica de complexidade e os seguintes valores serão reportados nesse trabalho (NACCARI *et al.*, 2015):

- Aceleração da codificação (AC): Calculada como a média aritmétrica dos valores medidos para cada um dos quatro valores de QP distintos conforme definido pela Equação 5.1. Valores positivos expressam uma redução na complexidade computacional e valores negativos expressam um retardo no processo de codificação.

$$\Delta T = \frac{1}{4}\sum_{i=1}^{4}\frac{T_{HM}(QP_i) - T_{\Omega}(QP_i)}{T_{HM}(QP_i)} \times 100 \tag{5.1}$$

em que $T_{HM}$ representa o tempo de processamento usando a implementação de referência HM e $T_{\Omega}$ representa o tempo de processamento usando a versão modificada com as contribuições desta tese.

- Variação do tempo de aceleração ($\Delta T$): Calculada como a diferença percentual entre o tempo de aceleração máximo e o mínimo. Essa medida indica a variação do tempo de codificação dos pontos medidos. Quanto menor for esse valor, mais eficiente é o algoritmo proposto para diferentes aplicações de taxa de transmissão distintas.

## 5.4 O Algoritmo Proposto – O Codificador Ω

A análise feita por (ZUPANCIC *et al.*, 2015a) mostrou que a etapa mais dispendiosa para a codificação de conteúdos UHD com o codificador HEVC são a decisão do modo de divisão da unidade de codificação e a predição inter devido ao processo de estimação de movimento.

As otimizações de codificação propostas irão abordar a etapa RDO (*Rate Distorcion Optimization*) no modo inter-quadros em todos os particionamentos de uma unidade de codificação. Primeiramente, um conjunto de dados de 15 atributos das unidades de codificaçao foram coletados considerando as características do processo de exploração de redundância temporal (inter-quadros) formando um vetor de 16 elementos, 15 atributos e 1 elemento de classificação, para cada unidade de codificação. Em cada unidade de codificação, e em cada profundidade, estas 15 características foram coletados junto com a classificação se a unidade de codificação atual é codificada usando o modo de particionamento $2N\times2N$ ou não. O modo de particionamento $2N\times2N$ define que este conjunto de dados composto por dezesseis elementos será posteriormente analisado em um processo de seleção de atributos usando a ferramenta Weka (WITTEN; FRANK, 2005), seguindo as recomendações do trabalho publicado em (CORREA *et al.*, 2015). Sete atributos foram inicialmente selecionados e, em seguida, as simulações foram realizadas usando duas sequências para definir o conjunto mínimo de atributos recurso para ser usado com base na taxa de erro coletada em treinos de classificação. O erro de classificação foi 11,46% e diminui para 0,9% usando três características. A Figura 5.1 ilustra como a taxa de erro descresceu com a diminuição do número de atributos.

O ordenamento foi feito usando o algoritmo *Information Gain Analyses Evaluation* (IGAE) fornecido pelo Weka. Os três atributos selecionados representam, respectivamente, a informação se as unidades de predição posicionadas à esquerda e na posição superior dos quadrantes foram codificadas no modo Skip, para os quadrantes 1, 2 e 3. O método implementado retorna zero se nenhuma das posições foi codificada como Skip e 1 se uma das duas posições, ou as duas, forem codificadas como Skip. Dessa forma, a amostra de treinamento utilizada é composta por esses três atributos selecionados pelo IGAE e pela informação se o particionamento selecionado foi o formato $2N\times2N$ ou não, caracterizando o problema abordado como uma classificação binária.

**Tabela 5.1** Descrição dos atributos analisados no processo de seleção automática.

| ìndice | Nome da Variável ou Método | Descrição |
|---|---|---|
| 1 | TotalCost | Custo do processo RDO da unidade de codificação corrente |
| 2 | CUPelX | Coordenada X da unidade de codificação corrente em relação ao tamanho original da imagem |
| 3 | CUPelY | Coordenada Y da unidade de codificação corrente em relação ao tamanho original da imagem |
| 4 | CtuRsAddr | Número sequencial da unidade de codificação dentro de um quadro |
| 5 | TotalBits | Número total de bits associado |
| 6 | TotalDistortion | Distorção calculada ao fim do processo RDO |
| 7 | MergeFlag | Determinada se é um bloco fundido ou não |
| 8 | PredictionMode | Modo de predição |
| 9 | CtxSkipFlag0 | Método que verifica se as unidades de predição superior e superior esquerda da unidade de codificação vizinha do quadrante 0 tem valor 1 para sua variável SkipFlag |
| 10 | CtxSkipFlag1 | idem ao anterior, para o quadrante 1 |
| 11 | CtxSkipFlag2 | idem ao anterior, para o quadrante 2 |
| 12 | CtxSkipFlag3 | idem ao anterior, para o quadrante 3 |
| 13 | MergeAMP | Informa se a funcionalidade AMP está ativa ou não |
| 14 | Depth | Profundidade de divisão da unidade de codificação (0-3) |
| 15 | SkipFlag | Informa se o bloco corrente vai ser o último nível codificado |

Um modo de treinamento baseado no tamanho do GOP foi implementado. Somente o primeiro quadro de um GOP é utilizado para o treinamento do modelo e o algoritmo de classificação é executado nos quadros restantes do mesmo GOP.

Após a etapa de análise dos atributos mais relevantes e com a definição do vetor de amostras de treinamento, o classificador foi implementado e integrado dentro do código do HM. Como o processo de classificação se aplica ao particionamento da unidade de codificação, a implementação foi inserida no método que faz a avaliação da distorção e do particionamento. Esse método é o núcleo de toda análise RDO no codificador. O fluxo do algoritmo implementado está representado na Figura 5.2.

**Tabela 5.2** Atributos elencados em ordem de relevância segundo o algoritmo IGAE.

| Valor IGAE | Índice | Atributo (Nome da Variável) |
|---|---|---|
| 0.99315 | 14 | CtxSkipFlag3 |
| 0.89980 | 12 | CtxSkipFlag1 |
| 0.89864 | 13 | CtxSkipFlag2 |
| 0.74312 | 5 | TotalBits |
| 0.73920 | 7 | MergeFlag |
| 0.67420 | 11 | CtxSkipFlag0 |
| 0.59042 | 8 | PredictionMode |
| 0.31867 | 9 | Height |
| 0.31867 | 10 | Width |
| 0.31867 | 16 | Depth |
| 0.21417 | 15 | MergeAMP |
| 0.20512 | 1 | TotalCost |
| 0.07516 | 6 | TotalDistortion |
| 0.02681 | 4 | CtuRsAddr |
| 0.02539 | 3 | CUPelY |
| 0.00780 | 2 | CUPelX |

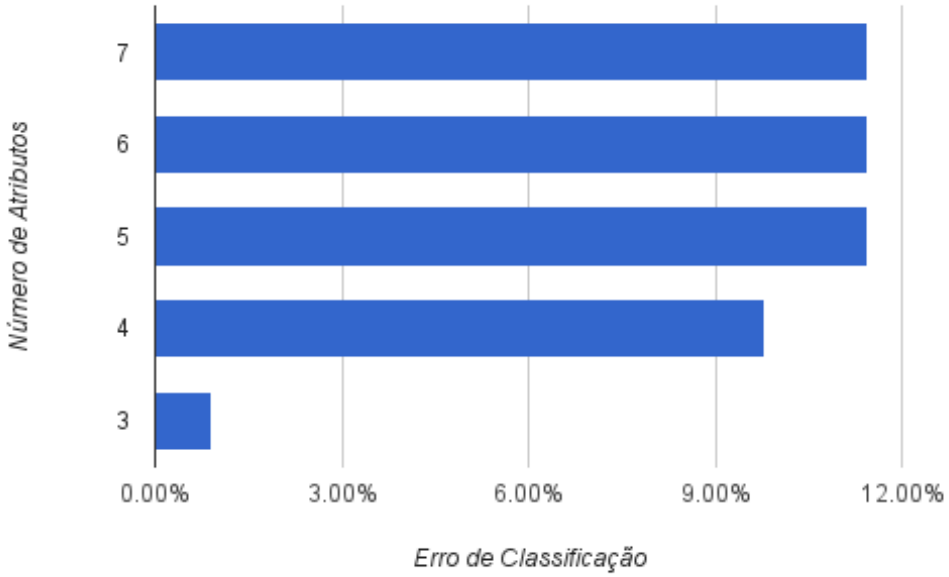



**Figura 5.1** Comportamento do erro de classificação com a redução do número de atributos

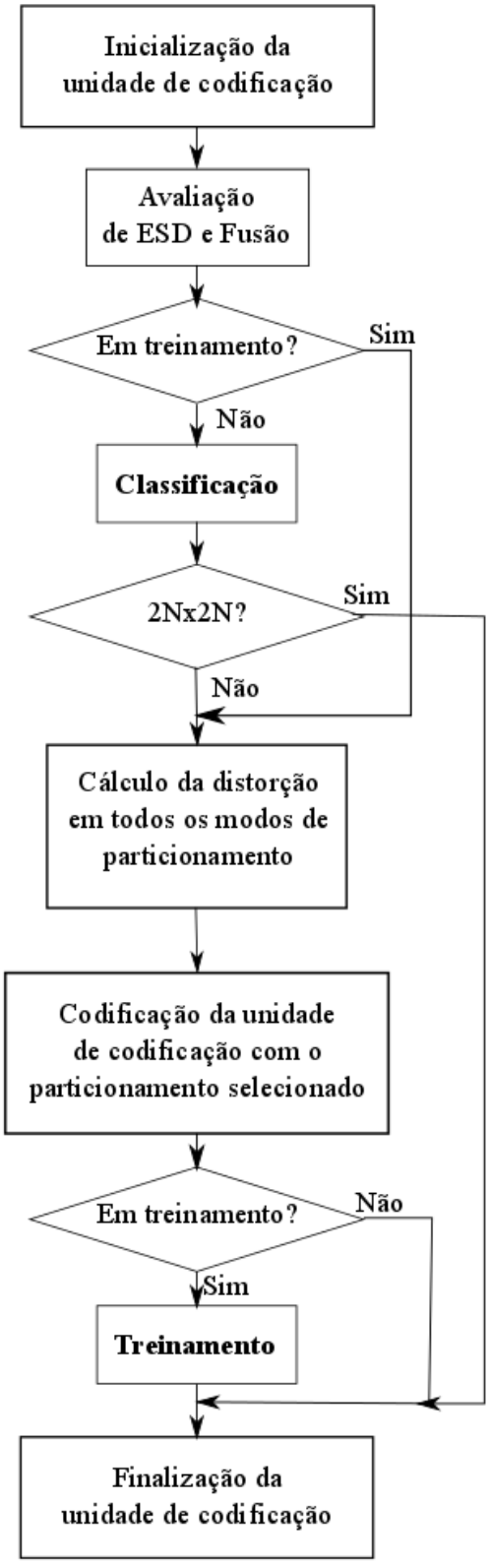



**Figura 5.2** Fluxograma do algoritmo proposto.

# CAPÍTULO 6

# Simulações e Resultados

## 6.1 Sequências de Vídeo Utilizadas

O conjunto de vídeos de testes é composto por dez vídeos UHD (3840×2160) e dois vídeo em HD (1920×1080) que serão utilizados para analisar o efeito do algoritmo proposto com a dimininuição da resolução do quadro. Todos os vídeos utilizados possuem formato 4:2:0, 60 quadros por segundo e oito *bits* por componente de informação. Considerando o formato de subamostragem de crominância 4:2:0, o vídeo resultante utiliza 16 *bits* por *pixel*. Os nomes e uma breve caracterização das sequências de vídeo estão listadas na Tabela 6.1.

As simulações foram realizadas usando os parâmetros especificados na Tabela 6.2 em um computador com quatro núcleos de processamento com arquitetura para suportar 8 processos em paralelo. A frequência de operação dos processadores de 2,8 GHz e a placa-mãe estava equipada com 24 GB de memória RAM com a tecnologia DDR3. Um quadro de cada sequência de vídeo usado nas simulações esta exibidos na Figura 6.1. Originalmente cada vídeo possuia 600 quadros, mas apenas 130 foram usados nas simulações. O principal motivo da redução do número de quadros foi devido ao elevado tempo de processamento, que em alguns casos, mesmo usando os 130 primeiros quadros de cada vídeo de teste, o tempo de simulação excedeu 10 horas para cada ponto do gráfico de cada configuração simulada. Por exemplo, o tempo de simulação total para o vídeo Campfire Party usando as configurações *Low Delay B* utrapassaria 51 horas, se os oito pontos fossem executados sequencialmente. Dado à capacidade do processador utilizado, quatro instâncias do processo do codificador eram invocadas paralelamente, reduzindo o tempo ao intervalo obtido do ponto de simulação com parâmetro de quantização igual a 22. Quanto menor o valor desse parâmetro, maior o tempo de simulação. Com essa paralelização, o tempo de simulação, em média, durava aproximadamente 15 horas para cada vídeo. Essa quantidade de quadros também se mostrou suficiente para verificar a estabilização do patamar do erro de classificação para a maioria dos vídeos testados. Em linhas gerais, o processo de simulação consiste em:

**Tabela 6.1** Descrição do conjunto de vídeos utilizados.

| **Nome** | **SI** | **TI** | **Resolução** | **Descrição** |
|---|---|---|---|---|
| Kimono | | | HD | Movimentação de cenário ao fundo com câmera focada no rosto de uma mulher. |
| BasketballDrive | | | HD | Rápida movimentação de uma partida de basquete. |
| Library | | | UHD | Eestudantes andando na frente de uma biblioteca. |
| Wood | | | UHD | Câmera movendo-se entre árvores com forte presença de raios solares. |
| Fountains | | | UHD | Movimentação de água em uma fonte artificial. |
| Construction Field | | | UHD | Máquinas escavadoras durante o trabalho. |
| Rush Hour | | | UHD | Movimentação de pessoas durante horário de pico. |
| Tall Buildings | | | UHD | Uma paisagem de edifícios altos. |
| Campfire | | | UHD | Uma fogueira. |
| Tree Shade | | | UHD | Exibe a sombra de uma grande arvóre com pouca movimentação de câmera e de fundo. |
| Runners | | | UHD | Vários corredores. Fundo parado e pessoas correndo. |

- Classificar os vídeos de acordo com os valores obtidos pelos cálculos da complexidade espacial (SI) e temporal (TI).

- Codificar cada vídeo quatros vezes em cada codificador, totalizando 8 execuções.

  Cada simulação de um codificador usa um parâmetro de quantização diferente seguindo os valores 22, 27, 32 e 37, conforme as recomendação em JCTVC L.1100. Essa faixa de valores garante uma boa variação da qualidade do vídeo resultante.

- Em cada simulação são registrados os valores da PSNR resultante da componente de luminância, crominância azul e crominância vermelha e também são coletados os valores da taxa de *bits* resultante e o tempo total de processamento.

- Sendo $QP_i = 22, 27, 32 e 37$, e com os valores de taxa de *bits* e relação sinal-ruído de pico da componente de luminância, obtem-se o gráfico da relação taxa/distorção exibindo as duas curvas com os valores de

  $(taxa_{HM}(QP_i), YPSNR_{HM}(QP_i))$ para a curva do codificador HM e

  $(taxa_{\Omega}(QP_i), YPSNR_{\Omega}(QP_i))$ para a curva para o codificador proposto $\Omega$.

(a) CampFire.

(b) Construction Field.

(c) Fountains.

(d) Library.

(e) Runners.

(f) Rush Hour.

(g) Residential Building.

(h) Tree Shade.

(i) Kimono.

(j) Basketball Drive.

**Figura 6.1** Exemplos de quadros dos vídeos utilizados nos testes.

## 6.2 A Parametrização dos Codificadores

Uma configuração de codificação e ambiente de teste bem definidos precisam ser estabelecidos para a realização das avaliações de desempenho. Nesta seção será descrita a parametrização do codificador de referência HEVC usado nas investigações. Além disso, será apresentado os conjuntos de configurações pré-definidos.

## 6.3 Descrição dos Experimentos

A norma HEVC especifica várias novas ferramentas de codificação para cada módulo de uma arquitetura genérica de um codificador baseado em técnicas de exploração de redundância espacial e temporal. Assumindo um ambiente computacionalmente restrito, é importante entender a relação custo-benefício entre a eficiência de compressão e a complexidade do codificador associadas a cada ferramenta de codificação. Portanto, um conjunto de experimentos foi realizado em (NACCARI *et al.*, 2015) para quantificar esta relação. As ferramentas consideradas para os experimentos realizados nesse trabalhos estão listadas na Tabela 6.2 e foram baseadas no trabalho citado anteriormente. Para cada ferramenta, os indicadores de desempenho mencionados na Seção 5.3 são calculados entre o codificador de referência e o codificador $\Omega$.

**Tabela 6.2** Configurações das principais ferramentas do codificador HEVC.

| **Nome da ferramenta** | **Parâmetro usado** |
|---|---|
| Tamanho máximo CTU | 64×64 |
| Profundidade RQT | 3 |
| Tamanho máximo TU | 32×32 |
| *Transform Skip (TS)* | Habilitado |
| RDOQ | Habilitado |
| Codificação Intra | Padrão, 35 modos |
| Vetor de movimento com $\frac{1}{4}$ de precisão | Habilitado |
| Vetor de movimento com $\frac{1}{2}$ de precisão | Habilitado |
| *Asymmetric Motion Partition (AMP)* | Habilitado |
| Número de vetores candidatos à fusão | 5 |
| Número de quadros de referência | até 4 |
| *Early Skip Detection (ESD)* | Habilitado |
| *Coding Flag Mode* (CFM) | Desabilitado |

**Tabela 6.3** Tempo gasto em cada passo do codificador de referência.

| **Etapa** | **Tempo gasto (%)** |
|---|---|
| Etapa intra-quadros | 9 |
| Etapa inter-quadros | 76 |
| Outras etapas | 15 |

A algoritmo proposto, chamada de Ω, obteve uma aceleração competitiva para um valor baixo da métrica BD-Rate quando comparada com outras implementações de redução de complexidade computacional já publicadas e analisadas neste trabalho.

## 6.4 Simulações com a Parametrização *Low Delay B*

Conforme definido na Seção 5.1, o conjunto de parâmetros de configuração *Low Delay B* define um cenário de aplicações para vídeos de maior qualidade ou armazenamento. Os gráficos da maioria das simulações serão apresentados a seguida, descrevendo características pertinentes de cada simulação. Cada gráfico exibe o comportamento do erro de classificação em função no número de treinamentos realizados. É importante mencionar que o treinamento é feito em tempo de execução.

- A Figura 6.2 exibe o gráfico da simulação usando o vídeo *Campfire Party*. Eixo das ordenadas exibe os valores do erro de classificação que é percentual de vezes, para a quantidade de amostras já observadas, que o algoritmo de classificação (Pegasos) tem valor errado calculado durante a etapa de treinamento. A legenda no gráfico é referente aos valores do parâmetro de quantização (QP) utilizados nesta tese em conformidade com as recomendações do JCT-VC-L1100. Essa simulação apresentou valores mais altos para o erro de classificação, especialmente para o parâmetro de quantização com valor 22. Nessa simulação, as curvas referentes aos parâmetros de quantização 22 e 27 não convergiram dentro do intervalo de 130 quadros proposto.

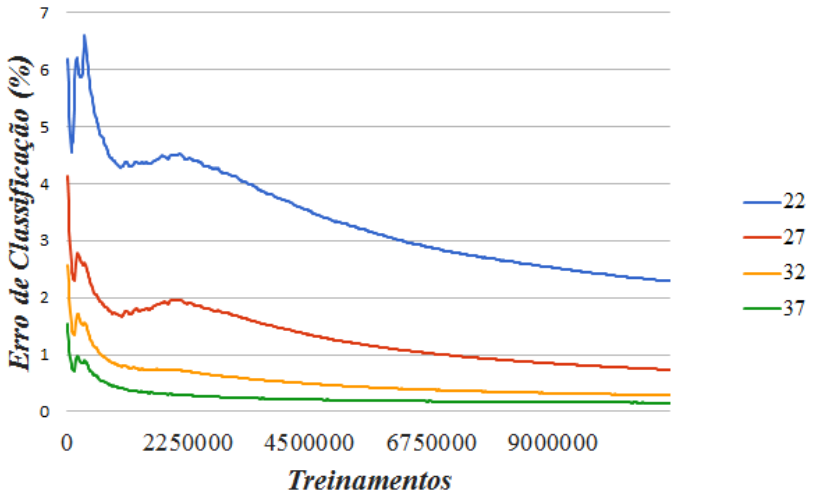



**Figura 6.2** Simulação para o vídeo Campfire Party usando a parametrização *Low Delay B*.

- Os resultados da simulação para o vídeo *Construction Field* mostram um baixo erro de classificação e uma rápida convergência. Todas as curvas convergem para um patamar

de erro de classificação semelhante, como pode ser observado na Figura 6.3. As simulações dos vídeos *Tree Shade* e *Wood* apresentaram as mesmas características e estão exibidas nas Figuras 6.5 e 6.4. No entanto, como pode-se observar, o erro de classificação nas simulações do vídeo Wood apresentou um patamar mais elevado, mas converge para valores aproximado aos das outras curvas. *Wood* é o vídeo com maior complexidade espacial e temporal dentro do conjunto de vídeos testados. Outro ponto importante é que, para esses vídeos, o erro de estabilização converge antes do último quadro usado, ou seja, dentro do intervalo de 130 quadros proposto nesse estudo.

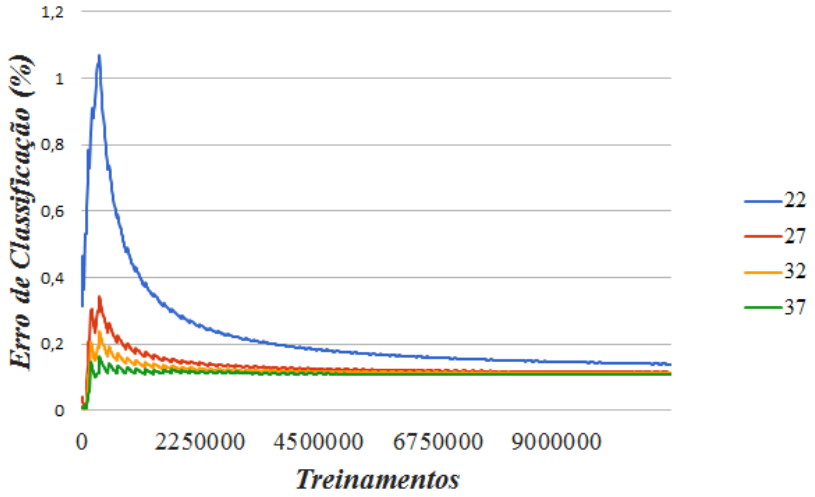



**Figura 6.3** Simulação para o vídeo Construction Field usando a parametrização *Low Delay B*.

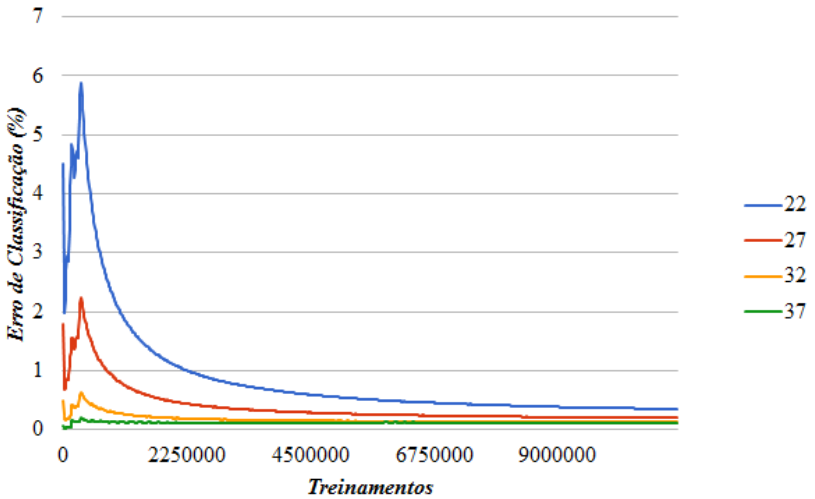



**Figura 6.4** Simulação para o vídeo Tree Shade usando a parametrização *Low Delay B*.

- Os resultados da simulação para o vídeo Fountains são exibidos na Figura 6.6. Destacam-se, nesse caso, o erro de classificação elevado apresentado nas primeiras amostras do

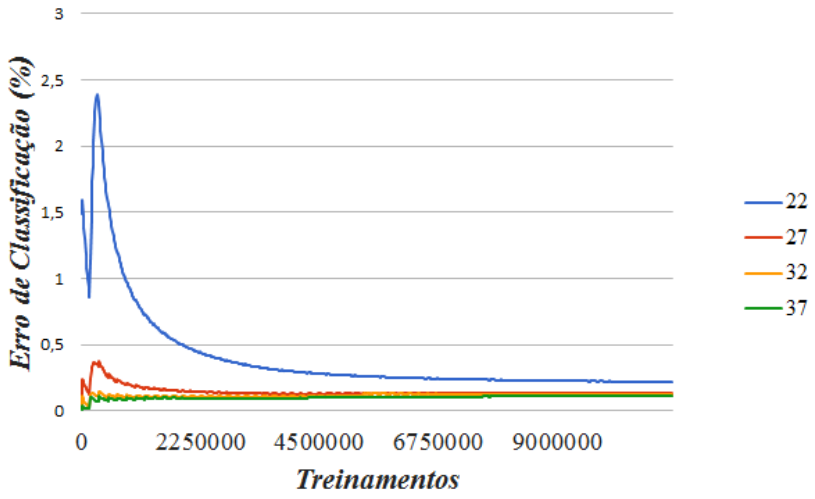



**Figura 6.5** Simulação para o vídeo Wood usando a parametrização *Low Delay B*.

treinamento para os valores 22 e 27 e o fato de as curvas convergirem para valores significativamente distintos.

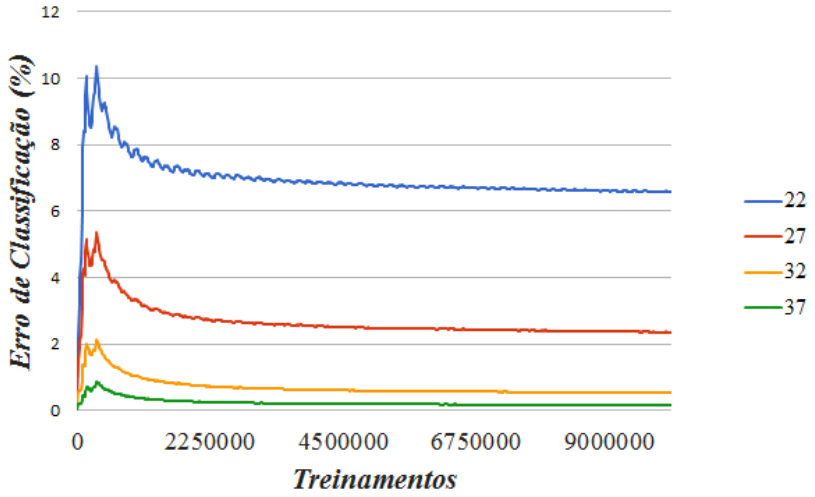



**Figura 6.6** Simulação para o vídeo Fountains usando a parametrização *Low Delay B*.

- As quatro curvas da Figura 6.7 são calculadas pela média aritmétrica de todas as simulações que usaram a parametrização *Low Delay B*. Na maioria das simulações, o erro de classificação convergiu para valores reduzidos. No entanto, como pode ser observado, a sequência *Fountains* apresentou um patamar mais elevado para o erro de classificação, o que influenciou no aumento da média.

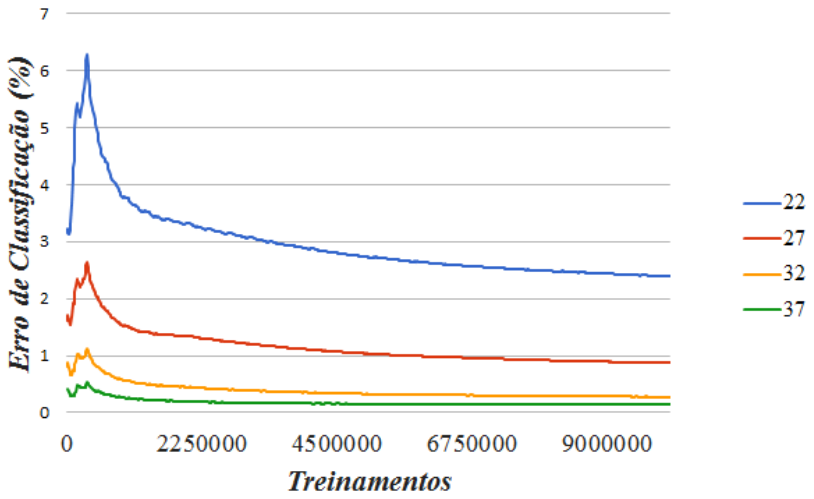



**Figura 6.7** Média dos valores dos erros de classificação obtidos nas simulações com a parametrização *Low Delay B*.

## 6.5 Simulações com a Parametrização *Random Access*

Conforme definido na Seção 5.1, o conjunto de parâmetros de configuração *Random Access* é utilizado para transmissões. A eficiência de codificação dessa configuração é maior do que as outras por fazer uso extensivo da predição bidirecional. Em contrapartida, isso aumenta o atraso de exibição. Para controlar a propagação de erros e facilitar o acesso ao vídeo, quadros I são inseridos periodicamente na sequência de codificação. Os gráficos de algumas simulações serão apresentados a seguida, descrevendo características pertinentes.

- Para validação do algoritmo, durante a fase de desenvolvimento, foram utlizadas duas sequências com resolução $1920 \times 1080$ conforme informado na Tabela 6.1. Ao finalizar a implementação do Codificador Ω chegou-se aos resultados da Figura 6.8. Nessa simulação foram usados 60 quadros, intervalo suficiente para verificar uma convergência do erro de classificação.

- Na Figura 6.9 são exibidas as curvas para cada parâmetro de configuração utilizado nas simulações usando a parametrização *Random Access* para o vídeo *Campfire Party*. Essa simulação apresentou um dos piores desempenhos, em especial para o valor 22 do parâmetro de quantização que, dentro do intervalo simulado, manteve-se acima de 10%. Essa sequência possui um elevado índice de complexidade temporal.

- A Figura 6.10 exibe as curvas das simulações com o vídeo *Construction Field*. Destaca-se nesse caso, o fato da curva do parâmetro de quantização 22 iniciar com erro de erro de classificação elevado e convergir para valores menores semelhante aos valores das outras curvas.

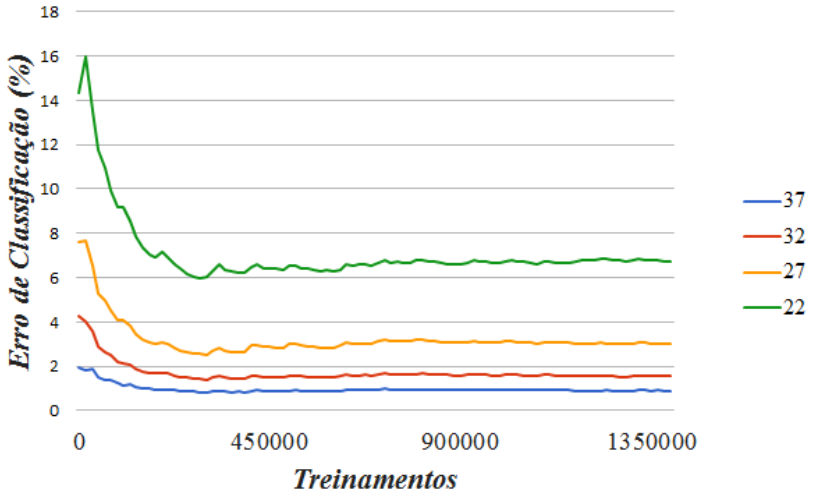



**Figura 6.8** Simulação para o vídeo BasketBall Drive usando a parametrização *Random Access* para 10 quadros.

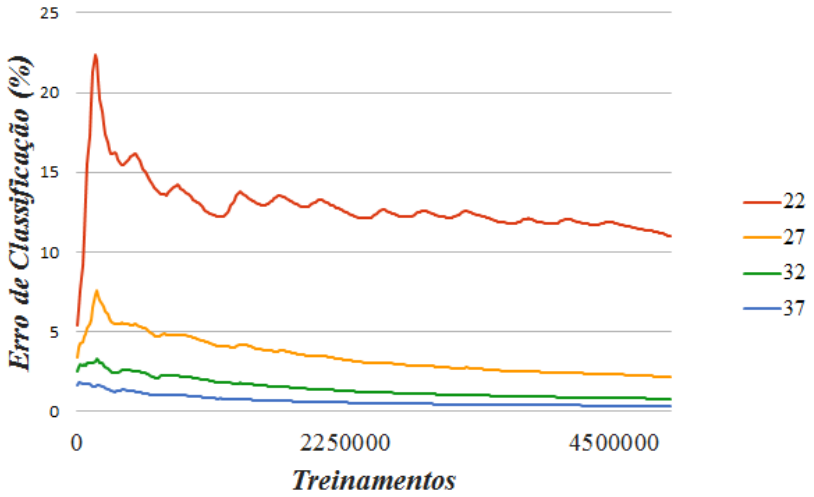



**Figura 6.9** Simulação para o vídeo Campfire Party Drive usando a parametrização *Random Access*.

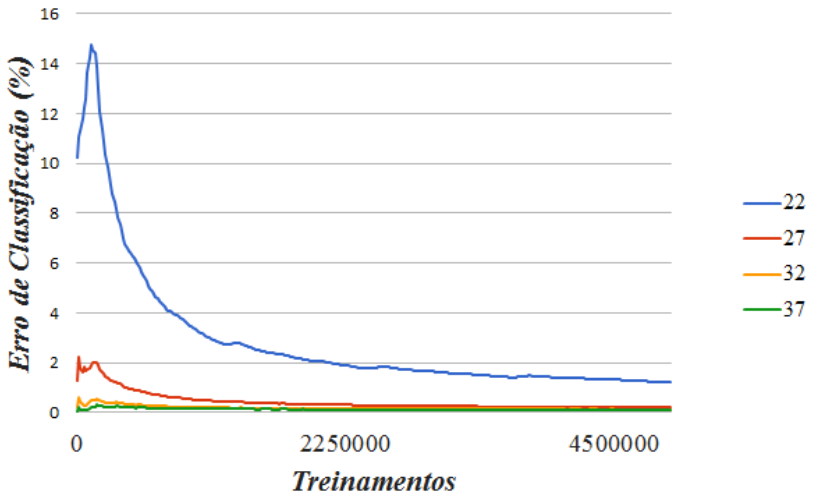



**Figura 6.10** Simulação para o vídeo *Construction Field* usando a parametrização *Random Access*.

- Assim como para o caso *Low Delay B*, o vídeo *Fountains* apresentou elevados patamar de erro elevado em relação às outras simulações. Essa sequência possui baixa complexidade espacial e temporal e sugere que outras características devem influenciar mais no modelo. Mesmo com o elevado erro, a eficiência de codificação foi aceitável, em relação aos outros trabalhos revisados, com uma redução de complexidade computacional de $29,82\%$

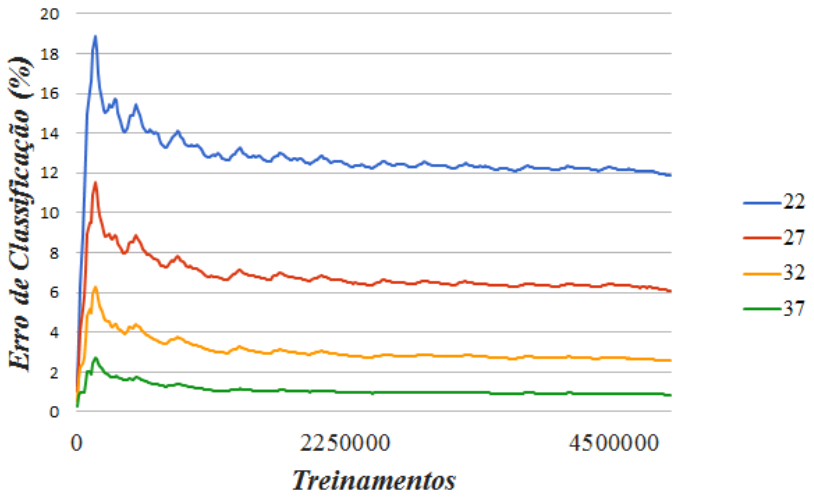



**Figura 6.11** Simulação para o vídeo *Fountains* usando a parametrização *Random Access*.

- A simulação para o vídeo *Library* no caso *Random Access* mostra uma rápida convergência de todas as curvas para valores similares na Figura 6.12. A curva do parâmetro de quantização 22 chega à $7\%$ mas cai para $0,5\%$ dentro do intevalo simulado de 130 quadros. A Figura 6.13 exibe as curvas separadas em eixos independentes para facilitar a visualização do comportamento das curvas com parâmetros de quantização maiores.

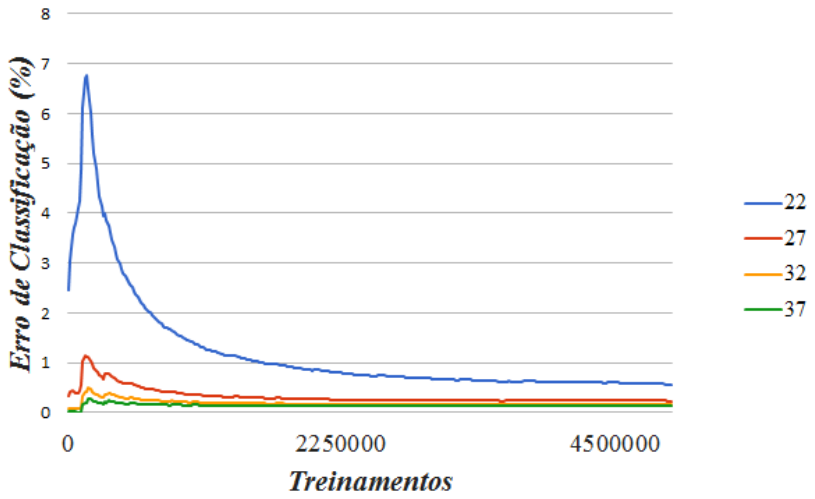



**Figura 6.12** Simulação para o vídeo Library usando a parametrização *Random Access*.

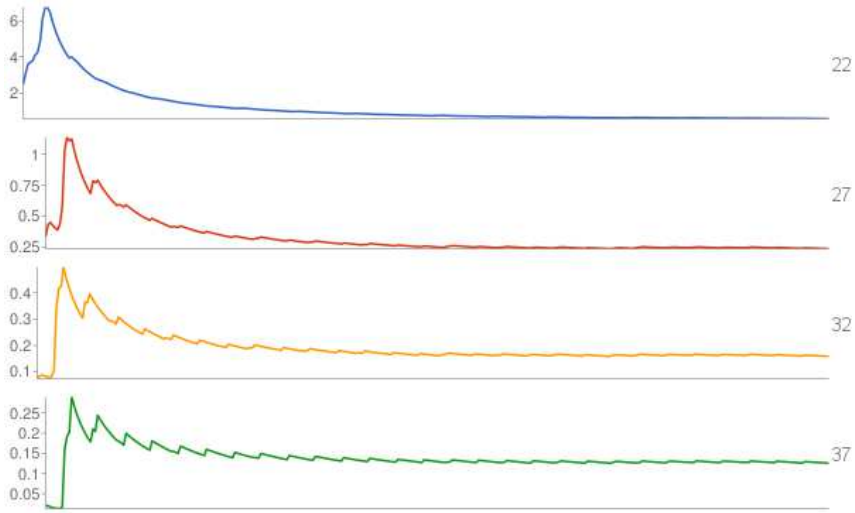



**Figura 6.13** Gráficos para o vídeo *Library* usando a parametrização *Random Access* em eixos independentes.

- A Figura 6.14 e a Figura 6.15 exibem as curvas das simulações para os vídeo *Rush Hour* e *Tree Shade*, respectivamente. Os comportamentos das suas simulações são semelhantes às simulações do vídeo *Library*, com valores de erro mais elevados.

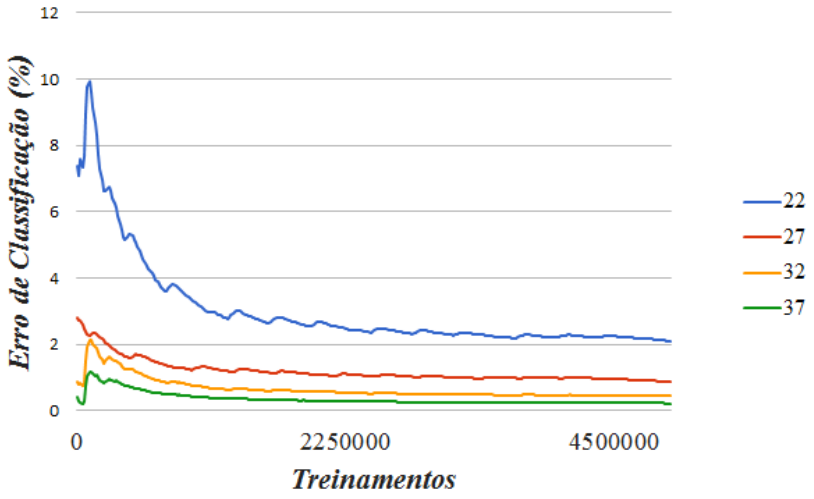



**Figura 6.14** Simulação para o vídeo *Rush Hour* usando a parametrização *Random Access*.

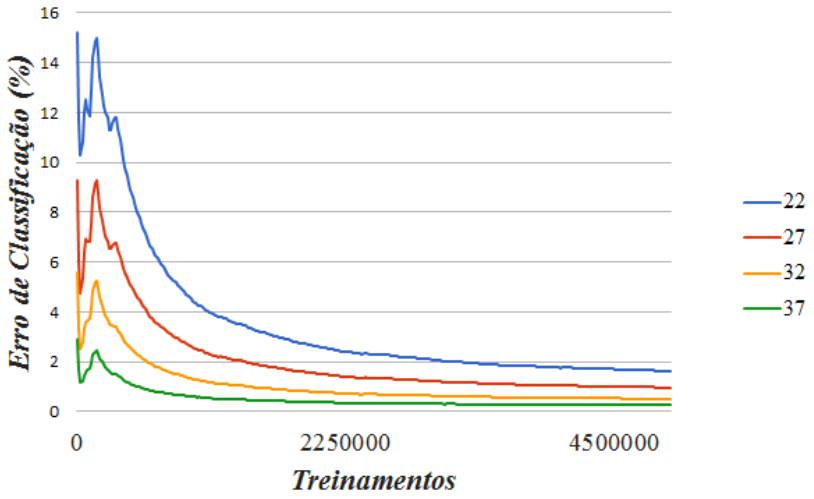



**Figura 6.15** Simulação para o vídeo *Tree Shade* usando a parametrização *Random Access*.

- As simulações do vídeo *Wood* são exibidas na Figura 6.16 e claramente acompanham a mesma tendência explicada para o caso *Low Delay B* com valores mais elevados para o erro de classificação.

- Na Figura 6.17 estão traçadas as curvas referentes às médias aritmétricas de todas as simulações com a parametrização *Random Access*.

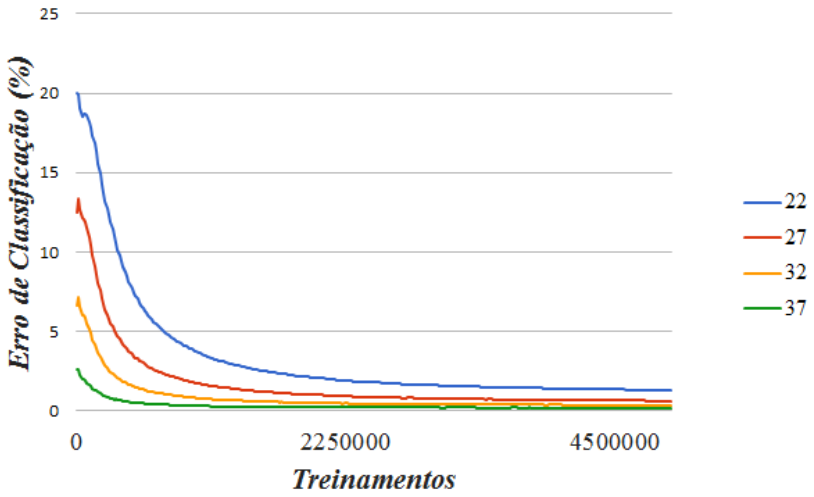



**Figura 6.16** Simulação para o vídeo *Wood* usando a parametrização *Random Access*.

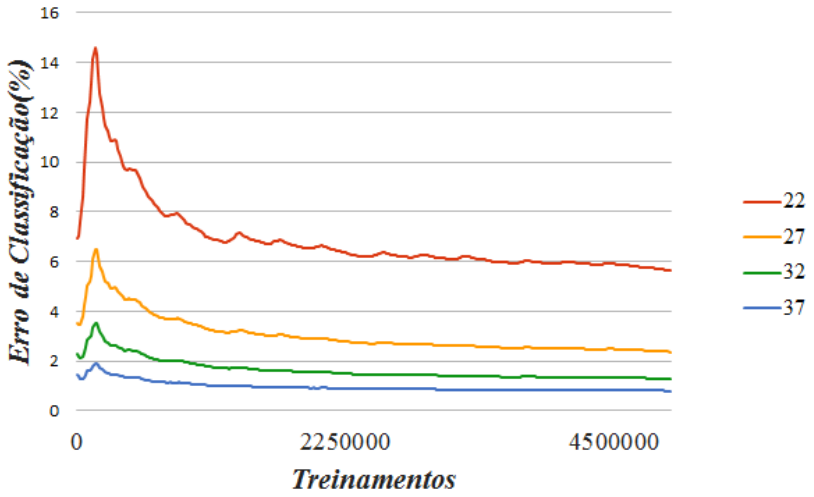



**Figura 6.17** Média dos valores dos erros de classificação obtidos nas simulações com a parametrização *Random Access*.

## 6.6 Simulações com Vídeos Concatenados

Uma das motivações principais do trabalho realizado nessa Tese foi propor um processo de otimização automática para codificação de vídeo digital em ultra alta definição. Considerando que uma sequência de vídeo comumente é composta de uma concatenação de cenas de diversas naturezas, com características espaço-temporais bem distintas, foi definido um caso de simulação composto pela concatenação de três dos vídeos utilizados. Foram gerados dois vídeos com conteúdos distintos e submetidos às mesmas simulações que os vídeos da base desse trabalho. O primeiro vídeo foi construído com a concatenação de 100 quadros do vídeo *Residential Building*, 50 quadros do vídeo *Runners* e 100 quadros do vídeo *Rush Hour*. O segundo vídeo foi construído com a concatenação de 80 quadros do vídeo *Wood*, 80 quadros do vídeo *Tree Shade* e 80 do vídeo *Fountains*.

Baseado nos resultados, as curvas no segundo intervalo no gráfico da Figura 6.18 e no terceiro intervalo no gráfico da Figura 6.19 não convergem. No entanto, o que de fato acontece, é que o erro de classificação do modelo esta voltando ao patamar de erro referente a cada vídeo. Pode-se observar também que o patamar inicial do erro de classificação nos segundos e terceiros intervalos não inicia com o mesmo valor das simulações independentes. Isso mostra que, o modelo previamente obtido no intervalo anterior apresenta-se como uma boa aproximação para o intervalo corrente. No decorrer dos quadros desse intervalo, o modelo de classificação gerado pelo algoritmo Pegasos será adaptado às características do conteúdo do mesmo.

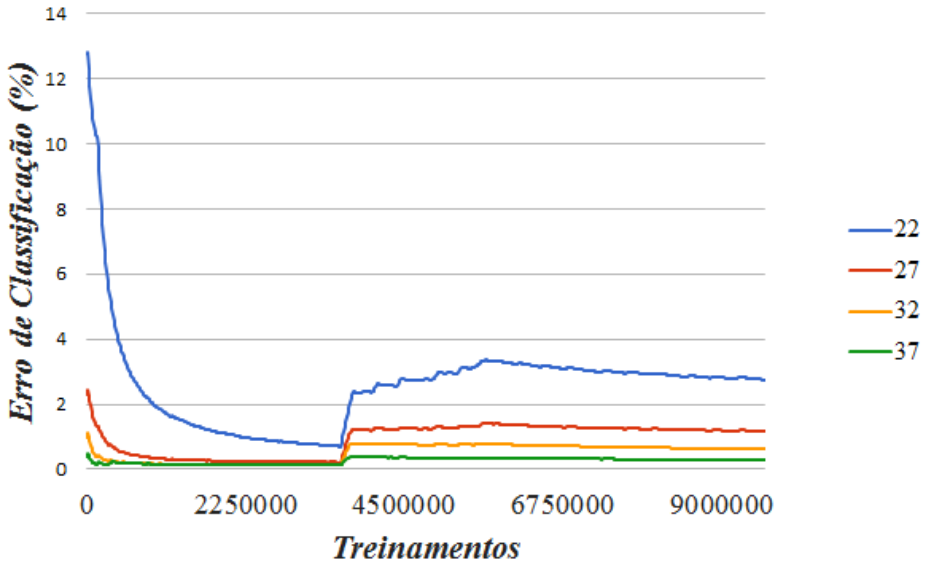



**Figura 6.18** Simulação do vídeo resultante da concatenação dos vídeos *Residential Building*, *Runners* e *Rush Hour*.

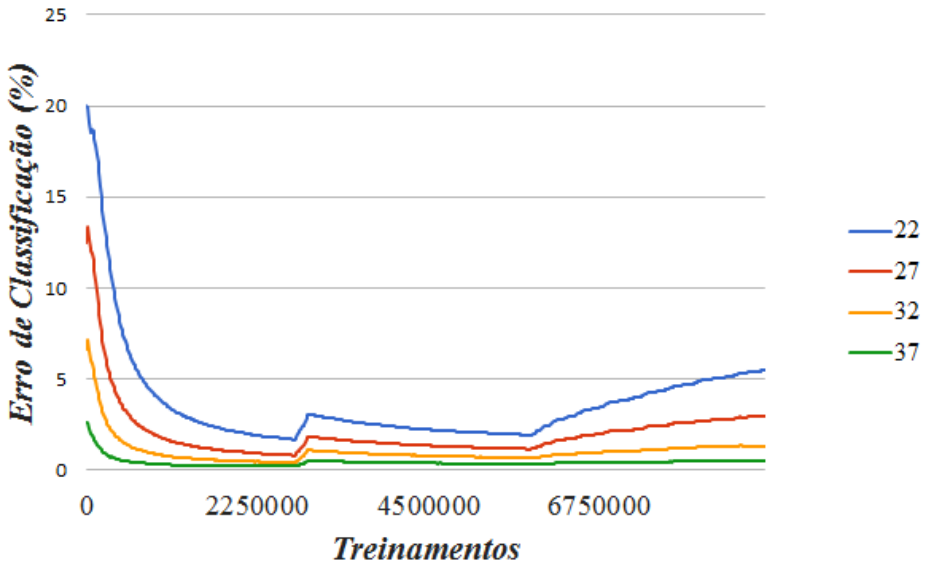



**Figura 6.19** Simulação do vídeo resultante da concatenação dos vídeos *Wood*, *Tree Shade* e *Fountains*.

## 6.7 Resultados

Nesta Seção serão discutidos os resultados alcançados, as conclusões e propostas de trabalhos futuros. A Figura 6.20 exibe um gráfico comparando a redução do tempo de processamento, usada para medir a complexidade computacional dos codificadores testados, para os quatro parâmetros de quantização recomendados pelo documento JCTVC L1100. As colunas do gráfico mostram que a redução do tempo de processamento é inversamente proporcional ao aumento do valor do parâmetro de quantização. Quanto menor esse valor, mais divisões no tamanho da unidade de codificação são necessárias para aumentar o nível de qualidade. Isso acontece acontece porque o limiar de distorção que é testado em cada divisão é propocional ao valor de QP. Dessa forma, o teste para verificar se a distorção da unidade de codificação corrente é suficientemente boa se torna mais rigoroso, fazendo com que o bloco se divida cada vez mais. Concluindo: um parâmetro de quantização menor implica em uma busca mais detalhada pelo melhor particionamento do CTU (*Coding Tree Unit*). Os resultados indicam uma redução de complexidade computacional competitiva com a maioria dos esquemas encontrados nos artigos revisados.

Para esclarecer a degradação inserida pelo processo proposto no Codificador Ω, a Figura 6.21 exibe o percentual de degradação da qualidade da componente de luminância, medida pela sua PSNR, em relação à mesma medida no codificador HM 16.7. O gráfico mostra que a perda de qualidade média é mínima para todas os parâmetros de codificação testados. Destaca-se o pequeno ganho (menor que $0,1\%$) obtido nas simulações com o QP igual a 22 para vídeos codificados com a parametrização *Low Delay B*.

As Tabelas 6.4 e 6.5 exibem os resultados detalhados de todas as simulações, considerando a taxa de *bits* e a relação sinal-ruído de pico da componente de luminância (YPSNR). A

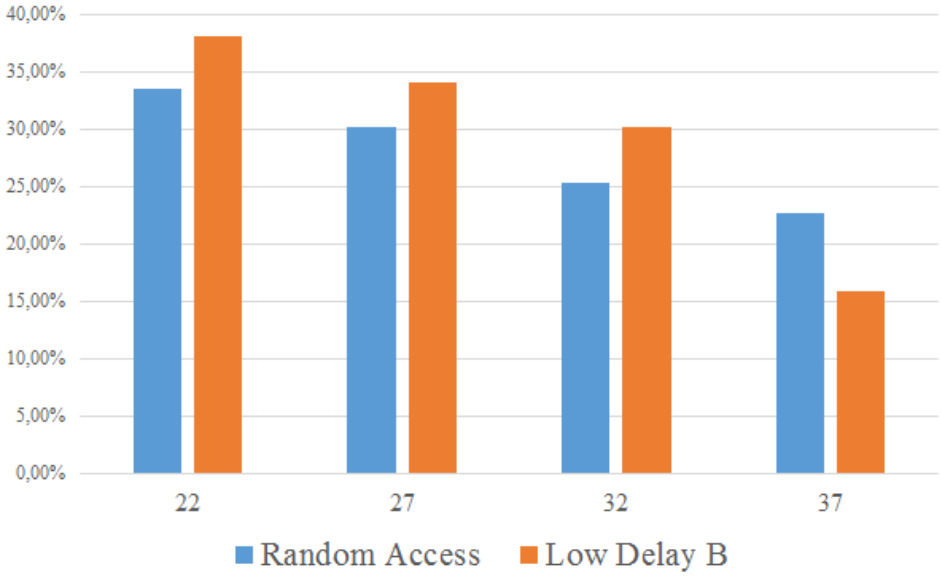



**Figura 6.20** Comparativo da redução de complexidade computacional para as parametrizações *Random Access* e *Low Delay B*.

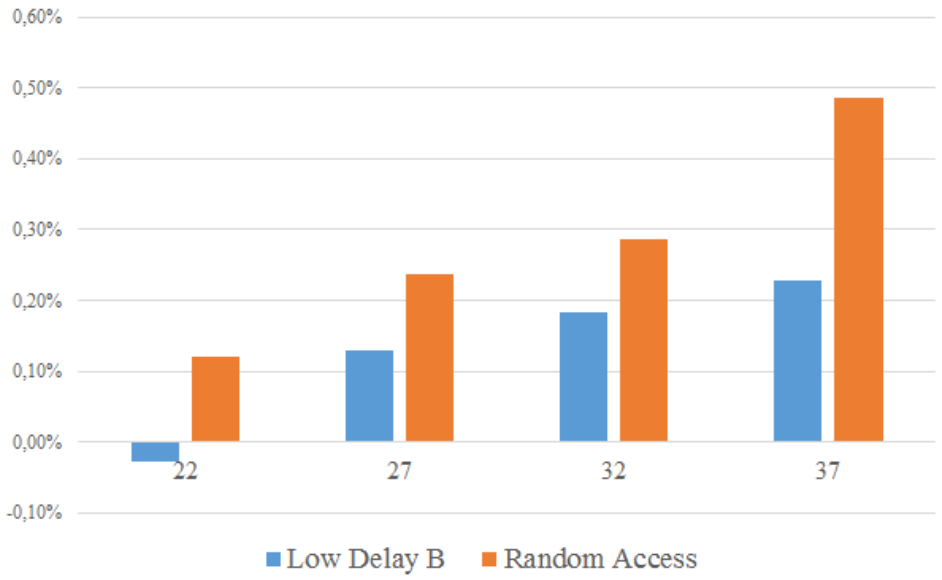



**Figura 6.21** Comparativo da degradação do vídeo, medida pela YPSNR, para as parametrizações *Random Access Low Delay B*.

média destes valores de YPSNR estão exibidos na Figura 6.21. Nestas tabelas pode-se verificar a semelhança dos resultados tanto para a taxa de *bits* como para a YPSNR.

**Tabela 6.4** Resultados da comparação com o HM 16.7 em termos de eficiência de compressão - Primeira parte

| **Vídeo** | **Aplicação** | **SI** | **TI** | **QP** | **Bitrate (kbps)** | | **YPSNR(dB)** | |
|---|---|---|---|---|---|---|---|---|
| | | | | | **HM** | **Ω** | **HM** | **Ω** |
| BasketBallDrive | Random Access | | | 22 | 10505 | 10522 | 39,27 | 39,25 |
| | | | | 27 | 3656 | 3754 | 37,65 | 37,6 |
| | | | | 32 | 1732 | 1779 | 35,83 | 35,74 |
| | | | | 37 | 920 | 944 | 33,86 | 33,74 |
| Library | Low Delay P | 6,4 | 21 | 22 | 5745 | 5649 | 41,01 | 41,01 |
| | | | | 27 | 1776 | 1739 | 39,88 | 39,97 |
| | | | | 32 | 817 | 806 | 38,35 | 38,84 |
| | | | | 37 | 429 | 425 | 36,46 | 36,45 |
| Library | Random Access | 6,4 | 21 | 22 | 7774 | 7877 | 41,19 | 41,2 |
| | | | | 27 | 2834 | 2871 | 40,23 | 40,3 |
| | | | | 32 | 1446 | 1460 | 38,8 | 38,77 |
| | | | | 37 | 795 | 800 | 36,99 | 36,95 |
| Fountains | Random Access | 5,5 | 13 | 22 | 70782 | 71128 | 40,05 | 40,03 |
| | | | | 27 | 30634 | 30821 | 36,99 | 36,8 |
| | | | | 32 | 12598 | 12667 | 34,01 | 33,97 |
| | | | | 37 | 5038 | 5072 | 31,78 | 31,73 |
| Fountains | Low Delay B | 5,5 | 13 | 22 | 81252 | 81447 | 40,74 | 40,73 |
| | | | | 27 | 37558 | 37690 | 37,69 | 37,69 |
| | | | | 32 | 15477 | 15540 | 34,67 | 34,66 |
| | | | | 37 | 6959 | 6126 | 32,6 | 32,22 |
| TallBuildings | Low Delay B | 8,2 | 17 | 22 | 14697 | 14859 | 40,28 | 40,24 |
| | | | | 27 | 4312 | 4357 | 37,99 | 37,97 |
| | | | | 32 | 1651 | 1659 | 35,5 | 35,48 |
| | | | | 37 | 726 | 731 | 32,84 | 32,83 |
| Construction Field | Low Delay B | | | 22 | 15414 | 15616 | 39,66 | 40,03 |
| | | | | 27 | 3177 | 3201 | 38,74 | 38,63 |
| | | | | 32 | 1596 | 1605 | 37,55 | 37,43 |
| | | | | 37 | 877 | 881 | 35,91 | 35,78 |

As Tabelas 6.6 e 6.7 listam todos os tempos de execução registrados durante as simulações.

As Tabelas 6.8 e 6.9 listam os resultados da medição da eficiência de codificação usando as métricas de Bjøtegaard desenvolvidas em (BJONTEGAARD, 2001), seguindo as orientações do documento JCTVC L1100 (BOSSEN, February, 2012). A Tabela 6.8 refere-se aos resultados das simulações com a pametrização *Random Access* e a Tabela 6.9 refere-se às simulações com a parametrização *Low Delay B*. Os resultados médios se mostraram compatíveis com os resultados dos trabalhos revisados, com destaque para o resultado da parametrização *Low Delay B* que apresentou uma degradação muito reduzida e até um aumento da qualidade medida para alguns vídeos. É importante enfatizar que as simulações foram realizadas com a otimização nativa ESD (*Early Skip Detection*) habilitada conforme listado na Tabela 6.2. A Tabela **??** mostra os resultados para dois vídeos codificados sem ESD, onde é percebido uma redução da complexidade computacional ainda maior.

**Tabela 6.5** Resultados da comparação com o HM 16.7 em termos de eficiência de compressão - Segunda parte

| **Vídeo** | **Aplicação** | **SI** | **TI** | **QP** | **Bitrate (kbps)** | | **YPSNR(dB)** | |
|---|---|---|---|---|---|---|---|---|
| | | | | | **HM** | **Ω** | **HM** | **Ω** |
| Construction Field | Random Access | 3,9 | 8,1 | 22 | 15669 | 15677 | 39,56 | 39,52 |
| | | | | 27 | 3197 | 3203 | 38,70 | 38,66 |
| | | | | 32 | 1596 | 1605 | 37,45 | 37,43 |
| | | | | 37 | 877 | 881 | 35,81 | 35,78 |
| Campfire Party | Low Delay B | 5,4 | 39,5 | 22 | 84003 | 84047 | 39,47 | 39,46 |
| | | | | 27 | 24585 | 24678 | 37,36 | 37,36 |
| | | | | 32 | 11538 | 11561 | 35,77 | 35,76 |
| | | | | 37 | 6520 | 6521 | 34,11 | 34,10 |
| Campfire Party | Random Access | 5,4 | 39,5 | 22 | 71766 | 72017 | 38,97 | 38,97 |
| | | | | 27 | 22839 | 22976 | 37,11 | 37,06 |
| | | | | 32 | 10209 | 10244 | 35,59 | 35,54 |
| | | | | 37 | 5861 | 5880 | 33,96 | 33,94 |
| Tree Shade | Random Access | 6,4 | 8,4 | 22 | 26516 | 27009 | 40,49 | 40,46 |
| | | | | 27 | 11621 | 11666 | 38,41 | 38,34 |
| | | | | 32 | 5529 | 5550 | 36,03 | 35,94 |
| | | | | 37 | 2720 | 2753 | 33,53 | 33,45 |
| Tree Shade | Low Delay B | 6,4 | 8,4 | 22 | 23366 | 23848 | 40,49 | 40,46 |
| | | | | 27 | 9392 | 9562 | 38,2 | 38,17 |
| | | | | 32 | 3992 | 4026 | 35,68 | 35,65 |
| | | | | 37 | 1704 | 1692 | 33,08 | 33,2 |
| Wood | Low Delay B | 10,4 | 39,8 | 22 | 35593 | 35669 | 39,56 | 39,54 |
| | | | | 27 | 11569 | 11627 | 37,16 | 37,14 |
| | | | | 32 | 4820 | 4859 | 34,59 | 34,57 |
| | | | | 37 | 2166 | 2172 | 31,82 | 31,8 |
| Wood | Random Access | 10,4 | 39,8 | 22 | 29390 | 29487 | 39,56 | 39,73 |
| | | | | 27 | 10480 | 10598 | 37,39 | 37,37 |
| | | | | 32 | 4559 | 4610 | 34,76 | 34,75 |
| | | | | 37 | 2086 | 2143 | 31,91 | 31,89 |

**Tabela 6.6** Diminuição da complexidade computacional em comparação com o codificador HM. Primeira Parte.

| **Vídeo** | Aplicação | **SI** | **TI** | **QP** | **Tempo(s)** | | | **ΔT(%)** | **AC(%)** |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | **HM** | **Ω** | **%** | | |
| BasketBallDrive | Random Access | | | 22 | 7959 | 4822 | 39,41 | 8,08 | 32,20 |
| | | | | 27 | 5816 | 3682 | 37,72 | | |
| | | | | 32 | 4622 | 3622 | 20,34 | | |
| | | | | 37 | 3817 | 2621 | 31,33 | | |
| Library | Low Delay P | 6,4 | 21 | 22 | 10536 | 7602 | 27,85 | 20,32 | 16,02 |
| | | | | 27 | 7252 | 5986 | 17,46 | | |
| | | | | 32 | 6343 | 5628 | 11,27 | | |
| | | | | 37 | 5902 | 5428 | 7,52 | | |
| Library | Random Access | 6,4 | 21 | 22 | 9501 | 7753 | 18,4 | 13,04 | 10,19 |
| | | | | 27 | 7470 | 6744 | 9,72 | | |
| | | | | 32 | 6864 | 6364 | 7,28 | | |
| | | | | 37 | 6591 | 6238 | 5,36 | | |
| Fountains | Random Access | 5,5 | 13 | 22 | 30021 | 20245 | 32,56 | 6,48 | 29,98 |
| | | | | 27 | 21987 | 15234 | 30,67 | | |
| | | | | 32 | 15768 | 11043 | 29,97 | | |
| | | | | 37 | 11304 | 8356 | 26,08 | | |
| Fountains | Low Delay B | 5,5 | 13 | 22 | 40302 | 21775 | 45,97 | 32,79 | 36,91 |
| | | | | 27 | 32186 | 17348 | 46,10 | | |
| | | | | 32 | 23607 | 13598 | 42,40 | | |
| | | | | 37 | 12963 | 11254 | 13,18 | | |
| TallBuildings | Low Delay B | 8,2 | 17 | 22 | 16325 | 9958 | 39,10 | 30,82 | 24,70 |
| | | | | 27 | 9765 | 6746 | 30,92 | | |
| | | | | 32 | 7233 | 5750 | 20,50 | | |
| | | | | 37 | 6099 | 5594 | 8,28 | | |
| Construction Field | Low Delay B | 3,9 | 8,1 | 22 | 16413 | 12751 | 22,31 | 17,55 | 12,12 |
| | | | | 27 | 7972 | 7972 | 14,43 | | |
| | | | | 32 | 6836 | 6836 | 6,98 | | |
| | | | | 37 | 6441 | 6441 | 4,77 | | |

**Tabela 6.7** Diminuição da complexidade computacional em comparação com o codificador HM. Segunda Parte.

| **Vídeo** | Aplicação | **SI** | **TI** | **QP** | **Tempo (s)** | | | **ΔT(%)** | **AC(%)** |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | **HM** | **Ω** | **%** | | |
| Construction Field | Random Access | 3,9 | 8,1 | 22 | 15592 | 9588 | 38,51 | 5,94 | 34,11 |
| | | | | 27 | 10383 | 6822 | 34,30 | | |
| | | | | 32 | 9226 | 6359 | 31,08 | | |
| | | | | 37 | 9096 | 9134 | 32,56 | | |
| Campfire Party | Low Delay B | 5,4 | 39,5 | 22 | 44920 | 24017 | 46,53 | 9,76 | 41,78 |
| | | | | 27 | 30601 | 17257 | 43,61 | | |
| | | | | 32 | 23583 | 14103 | 40,20 | | |
| | | | | 37 | 19622 | 12406 | 36,78 | | |
| Campfire Party | Random Access | 5,4 | 39,5 | 22 | 32063 | 19215 | 40,07 | 4,58 | 38,12 |
| | | | | 27 | 22571 | 13539 | 40,02 | | |
| | | | | 32 | 17367 | 10910 | 37,18 | | |
| | | | | 37 | 14760 | 9522 | 35,49 | | |
| Tree Shade | Random Access | 6,4 | 8,4 | 22 | 16902 | 10617 | 37,18 | 31,21 | 11,32 |
| | | | | 27 | 12235 | 8191 | 33,05 | | |
| | | | | 32 | 9812 | 6991 | 28,75 | | |
| | | | | 37 | 8779 | 6508 | 25,87 | | |
| Tree Shade | Low Delay B | 6,4 | 8,4 | 22 | 26447 | 15700 | 40,64 | 25,29 | 30,69 |
| | | | | 27 | 19632 | 12644 | 35,59 | | |
| | | | | 32 | 16359 | 16359 | 31,19 | | |
| | | | | 37 | 12692 | 12692 | 15,35 | | |
| Wood | Low Delay B | 10,4 | 39,8 | 22 | 25459 | 16087 | 38,81 | 14,24 | 30,90 |
| | | | | 27 | 14806 | 9637 | 34,91 | | |
| | | | | 32 | 10657 | 7471 | 29,30 | | |
| | | | | 37 | 8514 | 6592 | 22,57 | | |
| Wood | Random Access | 10,4 | 39,8 | 22 | 25844 | 17119 | 33,76 | 16,85 | 30,96 |
| | | | | 27 | 18557 | 12367 | 33,36 | | |
| | | | | 32 | 17629 | 10596 | 39,89 | | |
| | | | | 37 | 11719 | 9749 | 16,81 | | |

**Tabela 6.8** Resultados das métricas BD-*Rate* e BR-PSNR para a parametrização *Random Access*.

| **Vídeo** | **BD-Rate**(%) | **BD-PSNR**dB |
|---|---|---|
| Kimono | -0,05 | 0,015 |
| Library | 0,55 | -0,01 |
| Wood | 1,07 | -0,036 |
| Fountains | 2,6 | -0.08 |
| Construction Field | 3,1 | -0,04 |
| Campfire | 0,90 | -0,01 |
| Tree Shade | 3,04 | -0,093 |
| **Média** | **1,6** | **-0,036** |

**Tabela 6.9** Resultados das métricas BD-*Rate* e BD-PSNR para a parametrização *Low Delay B*.

| **Vídeo** | **BD-Rate**(%) | **BD-PSNR**(dB) |
|---|---|---|
| *Kimono* | | |
| *Library* | 0,160 | 0,019 |
| *Wood* | -0,67 | 0,01 |
| *Fountains* | 0,8 | -0,02 |
| *Construction Field* | -0,95 | -0,037 |
| *Tall Buildings* | 1,60 | -0,041 |
| *Campfire* | 0,47 | -0,008 |
| *Tree Shade* | 1,5 | -0,043 |
| **Média** | **0,55** | **-0,017** |

# CAPÍTULO 7

# Conclusão

Os trabalhos revisados geralmente usam classificadores simples ou uma combinação desses, nos quais o desempenho de classificação, que resultam em aprimoramentos na eficiência de codificação e na redução da complexidade computacional, não podem ser melhorados ou transformados em tempo de execução. Dessa forma, esses classificadores podem apresentar dificuldades de se adaptar a sequências de vídeo com mudanças de características espaço-temporais.

Aprimoramento na eficiência de compressão podem implicar em aumento do custo computacional, que já é bem elevado, principalmente quando aplicado à codificação de vídeos UHD. Nessa tese, um eficiente algoritmo para definir a divisão de uma unidade de codificação foi proposto para reduzir a complexidade computacional. A determinação da divisão da unidade de codificação foi inferida por um modelo criado em uma abordagem de aprendizagem automática em tempo real que foi integrada no código do codificador de referência. Para esse fim, foi integrado o algoritmo de classificação Pegasos. A determinação dos atributos de treinamentos também foi determinada por um processo automático de seleção. Resultados experimentais demonstraram que o algoritmo proposto reduz significantemente o tempo de codificação com uma pequena perda de eficiência de compressão, principalmente considerando o fato de ser um modelo gerado em tempo real. Os resultados foram competitivos com o da maioria dos artigos revisados. Comparativamente, poucos trabalhos utilizam essa abordagem, optando por treinamentos externos e uma posterior implantação do modelo gerado no código. Com isso, conclui-se que a esta tese atingiu os objetivos propostos, que foram:

- Propor um algoritmo de redução em tempo real da complexidade computacional da codificador HEVC usando vídeos em UHD.
- Manter ou minimizar as perdas de qualidade dos vídeos em relação ao codificador HM 16.7.
- Atingir reduções da complexidade computacional em patamares compatíveis com as mais recentes publicações. O algoritmo desta tese supera diversos resultados revisados. A Ta-

bela 7.1 elenca os principais resultados em relação com os resultados desta tese. É possível observar que o codificador Ω possui resultados inferiores apenas do que as soluções de técnicas combinadas. Existem mais duas etapas de classificação usando buscas exaustivas da mesma forma que a utilizada para determinar o tipo de particionamento da unidade de codificação. O mesmo algoritmo implementado nesta tese pode ser usado nessas outras duas etapas, melhorando os resultados de redução da complexidade computacional.

- Viabizar o uso de técnicas de aprendizagem automática de propósito genérico para diminuir a interfêrencia do usuário de sistemas de codificação de vídeo em tarefas de otimização.

**Tabela 7.1** Comparações com os principais resultados da literatura revisada.

| **Trabalho** | **BD-Rate(%)** | **ΔT(%)** | **Comentários** |
|---|---|---|---|
| Ivan Zupancic, 09/2015 | 2,2 | 72 | Combinação de 3 técnicas |
| Jinlei Zhang, 08/2015 | 4,1 | 77 | Combinação de 3 técnicas |
| G. Corrêa, 04/2015 | 1,35 | 63 | Combinação de 3 árvores de classificação |
| Yun Zhang, 07/2015 | 1,98 | 51,5 | Combina 2 classificadores SVM |
| ***Codificador* Ω** | ***1,7*** | ***54,73*** | ***Apenas uma técnica*** |

## 7.1 Considerações Finais

A área de estudo abordada nesta tese é promissora e desafiante. A codificação de vídeo digital tem recebido atenção especial, e com isso altos investimentos, devido à crescente tendência da utilização de serviços de vídeos em resoluções elevadas através da Internet. O primeiro projeto de pesquisa teve como objetivo o aprofundamento na arquitetura do codificador e a implementação de um mecanismo de otimização por meios da configuração dos parâmetros da estimação de movimento, em tempo de execução, em função da *PSNR* medida em cada quadro codificado. No segundo projeto de pesquisa foi implantado, dentro do codificador H.264, uma etapa de filtragem usando o gradiente de Sobel. O resultado dessa etapa de filtragem gerava uma métrica da complexidade espacial do trecho da imagem. Com essa informação era possível selecionar o algoritmo de busca mais adequado para cada situação. Essa técnica foi usada em vídeos 3D e observou-se um melhor desempenho para os quadros referentes a informação de profundidade. E, por último, no terceiro projeto foi realizada uma detalhada revisão da arquitetura do codificador de referência do padrão HEVC e uma avaliação experimental do desempenho do mesmo medida por métricas objetivas que comprovadamente mapeavam algumas características da visão humana. Foi feito um comparativo com o desempenho do H.264 usando a mesma métrica, com a finalidade de aferir a melhor capacidade do HEVC em endereçar essas características.

Durante a fase de revisão bibliográfica, a principal fonte de trabalhos utilizadas para a caracterização do estado da arte sobre técnicas de otimização do processo de codificação de vídeo HEVC foi o periódico *Transactions of Circuits and Systems for Video Technologies* do

IEEE. Foi observado que os membros do comitê JCTVC, de forma geral, utilizam esse periódico para publicação dos avanços na área.

As etapas de estudo e implementações no código fonte tiveram um nível de complexidade elevado em função do tamanho do sistema. O codificador de referência HM 16.7 possui 153 arquivos contendo 55430 linhas de código-fonte C/C++. Isso implica em uma alta probabilidade que qualquer modificação possa afetar outras partes do código. Para isso, foi necessária uma avaliação detalhada da arquitetura antes de implantar o algoritmo de aprendizagem automática proposto. Resumidamente, o processo de produção desta tese englobou as seguintes etapas:

- Revisão da fundamentação teórica sobre o padrão HEVC.
- Estudo dos principais algoritmos de aprendizagem de máquina e suas principais aplicações.
- Estudo sobre as métricas de avaliação de qualidade objetiva
- Revisão do estado da arte sobre otimização do processo de codificação HEVC. Essa revisão foi focada nos trabalhos publicados no periódico IEEE TCSVT (*Transactions of Circuits and Systems for Video Technology*).
- Simulações e testes com bibliotecas de aprendizagem automática.
- Integração da biblioteca no sistema de compilação do código de referência HM 16.7.
- Estudo da arquitetura do código fonte do HM 16.7 para receber as modificações propostas.
- Implementação da etapa de treinamento e da etapa de predição.
- Sincronização do sistema de controle de treinamento e predição de acordo com a posição do quadro dentro de um GOP.
- Depuração e otimização.
- Planejamento das simulações.

## 7.2 Propostas de Trabalhos Futuros

Durante o desenvolvimento dessa tese de doutorado várias oportunidades de pesquisa se mostraram viáveis e podem gerar resultados e contribuições significativas para área de pesquisa de codificação de vídeo. São elas:

- Avaliação da parametrização do kernel utilizado pelo algoritmo Pegasus SVM, variando os parâmetros $C$ e $\gamma$ para reduzir a taxa de erro de classificação.

- Aumentar o tempo de treinamento para seleção de atributos, aumentando a quantidade de vídeos utilizados.
- Implementação de esquema de predição do tipo de particionamento intra-quadros utilizado. Nesse trabalho apenas o processo Inter foi explorado.
- Usar o algoritmo Pegasos para determinar o resultado do processo de estimação de movimento.
- Combinar as técnicas mencionadas acima em um mesmo codificador.
- Implementação de um sistema de simulação automatizado seguindo as recomendações do documento JCTVC L.1100.
- Implantação do algoritmo de detecção de mudança de cena para ativação da etapa de treinamento.
- Realizar as simulações usando os vídeos de outras resoluções recomendados no JCTVC L.1100.
- Realizar as simulações com ESD desabilitado.

## 7.3 Produção Bibliográfica

1. ALENCAR, M. S. ; OLIVEIRA, J. F. F. REGIS, C. D. M. Televisão em Três Dimensões. Capítulo em TV Digital. 2 ed. São Paulo: Editora Érica Ltda., 2012, v. 1. *http://www.erica.com.br/*

2. RTIC – Revista de Tecnologia da Informação e Comunicação. Padrão HEVC - Novas Tecnologias para Aplicações de Elevadas Taxas de Compressão de Vídeo Jean F. F. de Oliveira, Marcelo S. de Alencar DOI: *http://dx.doi.org/10.12721/2237-5112.v04n02a08*

3. *Performance Evaluation of the PWSSIM Metric for HEVC and H.264*. Journal of Procedia Computer Science by ELSEVIER (ISSN:1877-0509) Jean Felipe F. De Oliviera, Carlos Danilo M. Regis, Marcelo Sampaio de Alencar. Aceito em março 2015.

4. *An Online Learning Early Skip Decision Method for the HEVC Inter Process using the SVM-Based Pegasos Algorithm*. Oliveira, Jean Felipe; Alencar, Marcelo. IET Eletronics Letters, Volume 52, Edição 14, p. 1227-1229. Aceito para publicação em 07/07/2016.

# Referências Bibliográficas


AHN, S.; LEE, B.; KIM, M. A Novel Fast CU Encoding Scheme Based on Spatiotemporal Encoding Parameters for HEVC Inter Coding. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 25, n. 3, p. 422–435, March 2015. ISSN 1051-8215.

ALENCAR, M. S. *Televisão Digital*. São Paulo: Editora Érica, 2007.

ALVAREZ-MESA, M. *et al.* Parallel Video Decoding in the Emerging Standard. In: *ICASSP*. [S.l.: s.n.], 2012. p. 1545–1548. ISBN 9781467300469.

BARJATYA, A.; MEMBER, S. Block Matching Algorithms For Motion Estimation. *Search*, p. 1–6, 2004.

BARONCINI J. OHM, R. Report on Preliminary Subjective Testing of HEVC Compression Capability. *ITU-T/ISO/IEC Joint Collaborative Team on Video Coding document JCTVC-H1004,*, San Jose, CA, 2012.

BJONTEGAARD, G. Calculation of Average PSNR Differences Between RD-curves. 2001.

BOSSEN, F. Common Test Conditions and Software Reference Configurations, document JCTVC-L1100. *JCT-VC*, February, 2012.

BRACAMONTE, J. *et al.* A Low Complexity Change Detection Algorithm. *Change*, p. 7–12, 2005.

CHEN, K.-C. *et al.* Applying Automatic Kernel Parameter Selection Method to the Full Bandwidth RBF Kernel Function for Hyperspectral Image Classification. In: *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. [S.l.: s.n.], 2014. p. 3442–3445.

CISCO. *Cisco Visual Networking Index: Forecast and Methodology, 2015-2020*. [S.l.], 2015.

CORPORATION, M.; WAY. Recent Developments in Standardization of High Efficiency Video Coding ( HEVC ). v. 7798, n. 7798, 2010.

CORREA, G. *et al.* Fast HEVC Encoding Decisions Using Data Mining. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 25, n. 4, p. 660–673, April 2015. ISSN 1051-8215.

CORREA, G. *et al.* Classification-based Early Termination for Coding Tree Structure Decision in HEVC. In: *21st IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. [S.l.: s.n.], 2014. p. 239–242.

DENG, X. *et al.* Complexity Control of HEVC Based on Region-of-Interest Attention Model. In: *Visual Communications and Image Processing Conference, 2014 IEEE*. [S.l.: s.n.], 2014. p. 225–228.

FANG, X. *et al.* Fast HEVC Intra Coding Unit Size Decision Based on an Improved Bayesian Classification Framework. In: *Picture Coding Symposium (PCS)*. [S.l.: s.n.], 2013. p. 273–276.

FARIAS, C. Q. Visual Quality Estimation Using Objective Metrics. p. 764–770, 2011.

FFMPEG, A complete, cross-platform solution to record, convert and stream audio and video. Https://www.ffmpeg.org/. Accessed: 2015-11-30.

FURNARI, A. *et al.* Generalized Sobel Filters for Gradient Estimation of Distorted Images. In: *IEEE International Conference on Image Processing (ICIP)*. [S.l.: s.n.], 2015. p. 3250–3254.

GROUP, M. S. Implementation of Fast Motion Estimation Algorithms and Comparison with Full Search Method in H . 264. *Journal of Computer Science*, v. 8, n. 3, p. 139–143, 2008.

HSIEH, P.-J. *et al.* An Automatic Kernel Parameter Selection Method for Kernel Nonparametric Weighted Feature Extraction With the RBF Kernel for Hyperspectral Image Classification. In: *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. [S.l.: s.n.], 2015. p. 1706–1709.

HU, Q. *et al.* Analysis and Optimization of x265 encoder. In: *Visual Communications and Image Processing Conference, 2014 IEEE*. [S.l.: s.n.], 2014. p. 502–505.

JIA, Y.; LIN, W.; KASSIM, A. Estimating Just-Noticeable Distortion for Video. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 16, n. 7, p. 820–829, July 2006. ISSN 1051-8215.

KAMOLRAT, B. *et al. 3D Motion Estimation for Depth Image Coding in 3D Video Coding*. 2009. 824–830 p. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5174461>.

KIM, J.; BAE, S.-H.; KIM, M. An HEVC-Compliant Perceptual Video Coding Scheme Based on JND Models for Variable Block-Sized Transform Kernels. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 25, n. 11, p. 1786–1800, Nov 2015. ISSN 1051-8215.

KORHONEN, J.; REITER, U.; UKHANOVA, A. Frame rate versus spatial quality: Which video characteristics do matter? In: *Visual Communications and Image Processing (VCIP), 2013*. [S.l.: s.n.], 2013. p. 1–6.

KOUIROUKIDIS, N.; EVANGELIDIS, G. The Effects of Dimensionality Curse in High Dimensional kNN Search. In: *15th Panhellenic Conference on Informatics (PCI)*. [S.l.: s.n.], 2011. p. 41–45.

LAMPERT, C. H. Machine Learning for Video Compression: Macroblock Mode Decision. In: *18th International Conference on Pattern Recognition, 2006. ICPR 2006.* [S.l.: s.n.], 2006. v. 1, p. 936–940. ISSN 1051-4651.

LANGFORD, J.; LI, L.; ZHANG, T. Sparse Online Learning via Truncated Gradient. *J. Mach. Learn. Res.*, JMLR.org, v. 10, p. 777–801, jun. 2009. ISSN 1532-4435. Disponível em: <http://dl.acm.org/citation.cfm?id=1577069.1577097>.

LEE, H. *et al.* Early Skip Mode Decision for HEVC Encoder With Emphasis on Coding Quality. *IEEE Transactions on Broadcasting*, v. 61, n. 3, p. 388–397, Sept 2015. ISSN 0018-9316.

LEE, J.-H. *et al.* Fast Intra Mode Decision Algorithm Based on Local Binary Patterns in High Efficiency Video Coding (HEVC). In: *IEEE International Conference on Consumer Electronics (ICCE)*. [S.l.: s.n.], 2015. p. 270–272.

LI, X. *et al.* Context-Adaptive Fast Motion Estimation of HEVC. In: *IEEE International Symposium on Circuits and Systems (ISCAS)*. [S.l.: s.n.], 2015. p. 2784–2787.

LIU, T. *et al.* Perceptual Quality Measurement of Video Frames Affected by both Packet Losses and Coding Artifacts Polytechnic Institute of New York University Thomson Corporate Research Brooklyn , NY Princeton , NJ. *New York*.

MAHMOUD, H. *et al.* A New Efficient Block-Matching Algorithm for Motion Estimation. *The Journal of VLSI Signal ProcessingSystems for Signal Image and Video Technology*, v. 42, n. 1, p. 21–33, 2006. ISSN 09225773. Disponível em: <http://www.springerlink.com/index/10.1007/s11265-005-4160-2>.

MALLIKARACHCHI, T.; FERNANDO, A.; ARACHCHI, H. Effective Coding Unit Size Decision Based on Motion Homogeneity Classification for HEVC Inter Prediction. In: *IEEE International Conference on Image Processing (ICIP)*. [S.l.: s.n.], 2014. p. 3691–3695.

MENON, A. K. Large-Scale Support Vector Machines: Algorithms and Theory. 2009.

NACCARI, M. *et al.* HEVC Coding Optimisation for Ultra High Definition Television Services. In: *Picture Coding Symposium (PCS), 2015*. [S.l.: s.n.], 2015. p. 20–24.

NALLURI, P.; ALVES, L.; NAVARRO, A. Improvements to TZ search motion estimation algorithm for multiview video coding. In: *International Conference on Systems, Signals and Image Processing - IWSSIP*. [S.l.: s.n.], 2012. v. 19, p. 388–391.

NIGHTINGALE, J.; WANG, Q.; GRECOS, C. HEVStream: A Framework for Streaming and Evaluation of High Efficiency Video Coding (HEVC) Content in Loss-prone Networks. *IEEE Transaction of Consumer Electronics*, v. 58, n. 2, p. 404–412, 2012.

OHM, J.-r.; SULLIVAN, G. J. High Efficiency Video Coding: The Next Frontier in Video Compression. n. December 2012, p. 152–158, 2013.

OHM, J.-r. *et al.* Comparison of the Coding Efficiency of Video Coding Standards – Including High Efficiency Video Coding ( HEVC ). *IEEE Transactionsof Circuits and Systems for Video Technologies.*, v. 22, n. 12, p. 1669–1684, 2012.

OHM, X. L.; WIEN, M.; JENS-RAINER. Rate-Complexity-Distorcion Evaluation for Hybrid Video Coding. In: *Multimedia and Expo (ICME), IEEE International Conference on*. Suntec City: [s.n.], 2010. p. 685–690. ISBN 9781424474936.

PAN, Z. *et al.* Early MERGE Mode Decision Based on Motion Estimation and Hierarchical Depth Correlation for HEVC. *IEEE Transactions on Broadcasting*, v. 60, n. 2, p. 405–412, June 2014. ISSN 0018-9316.

PAN, Z. *et al.* Early termination for TZSearch in HEVC Motion Estimation. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.: s.n.], 2013. p. 1389–1393. ISSN 1520-6149.

PODDER, P.; PAUL, M.; MURSHED, M. Efficient Coding Strategy for HEVC Performance Improvement by Exploiting Motion Features. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.: s.n.], 2015. p. 1414–1418.

POURAZAD, B. M. T. *et al.* HEVC : The New Gold Standard for Video Compression. *Consumer Electronics Magazine, IEEE*, v. 1, n. 3, p. 36–46, 2012. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6222536>.

REGIS, C. D. de M. *et al.* Design of Objective Video Quality Metrics Using Spatial and Temporal Informations. *IEEE Latin America Transactions*, v. 13, n. 3, p. 790–795, March 2015. ISSN 1548-0992.

REGIS, C. D. M. *Métrica de Avaliação Objetiva de Vídeo Usando a Informação Espacial , a Temporal e a Disparidade*. Tese (Doutorado) — Federal University of Campina Grande – UFCG, 2013.

SEGALL, A. *et al. Report of Results of the Joint Call for Proposals on Scalable High Efficiency Video Coding (SHVC)*. Shanghai, China: ISO/IEC JTC1/SC29/WG11.

SESHADRINATHAN, K. *et al.* Study of Subjective and Objective Quality Assessment of Video. *IEEE Transactions on Image Processing*, v. 19, n. 6, p. 1427–1441, June 2010. ISSN 1057-7149.

SHALEV-SHWARTZ, S.; SINGER, Y.; SREBRO, N. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In: *Proceedings of the 24th International Conference on Machine Learning*. New York, NY, USA: ACM, 2007. (ICML '07), p. 807–814. ISBN 978-1-59593-793-3. Disponível em: <http://doi.acm.org/10.1145/1273496.1273598>.

SHANNON, C. E. A Mathematical Theory of Communication. *The Bell System Technical Journal*, v. 27, n. 3, p. 379–423, July 1948. ISSN 0005-8580.

SHEN, L.; ZHANG, Z.; LIU, Z. Adaptive Inter-Mode Decision for HEVC Jointly Utilizing Inter-Level and Spatiotemporal Correlations. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 24, n. 10, p. 1709–1722, Oct 2014. ISSN 1051-8215.

STOCKHAMMER, T.; HANNUKSELA, M. M.; WIEGAND, T. H.264/AVC in wireless environments. *Circuits and Systems for Video Technology, IEEE Transactions on*, v. 13, n. 7, p. 657– 673, Julho 2003.

SUGAWARA, M.; CHOI, S.-Y.; WOOD, D. Ultra-High-Definition Television (Rec. ITU-R BT.2020): A Generational Leap in the Evolution of Television [Standards in a Nutshell]. *IEEE Signal Processing Magazine*, v. 31, n. 3, p. 170–174, May 2014. ISSN 1053-5888.

SULLIVAN, G. J. *et al.* Overview of the High Efficiency Video Coding. *IEEE Trans. Circuits Syst. Video Techn*, v. 22, n. 12, p. 1649–1668, 2012.

SZCZERBA, K. *et al.* Fast Compressed Domain Motion Detection in H.264 Video Streams for Video Surveillance Applications. *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, p. 478–483, 2009. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5279618>.

SZE MADHUKAR BUDAGAVI, G. J. S. V. *High Efficiency Video Coding (HEVC) Algorithms and Architectures*. [S.l.]: Springer International Publishing, 2014.

SZE MADHUKAR BUDAGAVI, G. J. S. V. *High Efficiency Video Coding (HEVC) Algorithms and Architectures*. [S.l.]: Springer International Publishing, 2014.

TAN, T. *et al.* Video Quality Evaluation Methodology and Verification Testing of HEVC Compression Performance. *IEEE Transactions on Circuits and Systems for Video Technology*, PP, n. 99, 2015. ISSN 1051-8215.

TOHIDYPOUR, H.; POURAZAD, M.; NASIOPOULOS, P. Online Learning-based Complexity Reduction Scheme for 3D-HEVC. *IEEE Transactions on Circuits and Systems for Video Technology*, PP, n. 99, p. 1–1, 2015. ISSN 1051-8215.

VAN, L. P. *et al.* Fast Transrating for High Efficiency Video Coding Based on Machine Learning. In: *20th IEEE International Conference on Image Processing (ICIP)*. [S.l.: s.n.], 2013. p. 1573–1577.

VANNE, J. *et al.* Comparative Rate-Distortion-Complexity Analysis of HEVC and AVC Video Codecs. *Circuits and Systems for Video Technology, IEEE Transactions on*, v. 22, n. 12, p. 1885–1898, 2012.

VIITANEN, M.; VANNE, J.; HäMäLäINEN, T. D. Complexity Analysis of Next-Generation HEVC Decoder. p. 20–23, 2013.

WANG, C.-C. *et al.* An Effective TU Size Decision Method for Fast HEVC Encoders. In: *International Symposium on Computer Consumer and Control (IS3C)*. [S.l.: s.n.], 2014. p. 1195–1198.

WANG, S. *et al.* Perceptual Video Coding Based on SSIM-Inspired Divisive Normalization. *IEEE Transactions on Image Processing*, v. 22, n. 4, p. 1418–1429, 2013. Disponível em: <http://dblp.uni-trier.de/db/journals/tip/tip22.html>.

WIEGAND, P. H. *et al.* Block Merging for Quadtree-Based Partitioning in HEVC. *IEEE Trans. Circuits Syst. Video Techn.*, v. 22, n. 12, p. 1720–1731, 2012.

WIEGAND, T. *et al.* Overview of the H.264/AVC Video Coding Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 13, n. 7, p. 560–576, July 2003. ISSN 1051-8215.

WITTEN, I. H.; FRANK, E. *Data Mining: Practical Machine Learning Tools and Techniques*. 2nd. ed. San Francisco: Morgan Kaufmann, 2005.

WOLFF, T. *et al.* H.264 Coding Artifacts and Their Relation to Perceived Annoyance. *Signal Processing*, n. Eusipco, p. 1–5, 2006.

XIONG, J. *et al.* Fast and Efficient Inter CU Decision for High Efficiency Video Coding. In: *IEEE International Conference on Image Processing (ICIP)*. [S.l.: s.n.], 2014. p. 3715–3719.

XIONG, J. *et al.* A Fast HEVC Inter CU Selection Method Based on Pyramid Motion Divergence. *IEEE Transactions on Multimedia*, v. 16, n. 2, p. 559–564, Feb 2014. ISSN 1520-9210.

YASAKETHU, S. L. P. *et al.* Analyzing Perceptual Attributes of 3D Video. v. 55, n. 2, p. 864–872, 2009.

YU, H.; WINKLER, S. Image Complexity and Spatial Information. In: *Fifth International Workshop on Quality of Multimedia Experience (QoMEX)*. [S.l.: s.n.], 2013. p. 12–17.

YU, T.; ZHANG, Y.; COSMAN, P. Classification Based Fast Mode Decision for Stereo Video Coding. In: *20th IEEE International Conference on Image Processing (ICIP)*. [S.l.: s.n.], 2013. p. 1724–1728.

ZENG ABDUL REHMAN, J. W. K.; WANG, Z.; . From H.264 to HEVC: Coding Gaing Predicted by Objective Video Quality Assessment Models. *2013 International Workshop on Video Processing and Quality Metrics for Consumer Electronics ( VPQM2013 )*, 2013.

ZENG, H.; MA, K.-K.; CAI, C. Fast Mode Decision for Multiview Video Coding Using Mode Correlation. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 21, n. 11, p. 1659–1666, Nov 2011. ISSN 1051-8215.

ZHANG, J.; LI, B.; LI, H. An Efficient Fast Mode Decision Method for Inter Prediction in HEVC. *IEEE Transactions on Circuits and Systems for Video Technology*, PP, n. 99, p. 1–1, 2015. ISSN 1051-8215.

ZHANG, Y. *et al.* Machine Learning-Based Coding Unit Depth Decisions for Flexible Complexity Allocation in High Efficiency Video Coding. *IEEE Transactions on Image Processing*, v. 24, n. 7, p. 2225–2238, July 2015. ISSN 1057-7149.

ZUPANCIC, I.; BLASI, S.; IZQUIERDO, E. Inter Prediction Optimisations for Fast HEVC Encoding of Ultra High Definition Content. In: *International Conference on Systems, Signals and Image Processing (IWSSIP)*. [S.l.: s.n.], 2015. p. 85–88.

ZUPANCIC, I.; BLASI, S.; IZQUIERDO, E. Multiple Early Termination for Fast HEVC Coding of UHD Content. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.: s.n.], 2015. p. 1419–1423.