Universidade Estadual de Campinas – UNICAMP

Faculdade de Tecnologia – FT

Estudo comparativo de padrões de compressão de vídeos

Alan Gabriel Godeny **Pedro Henrique de Souza Pozelli**

Limeira - SP

2020

Universidade Estadual de Campinas – UNICAMP

Faculdade de Tecnologia – FT

Estudo comparativo de padrões de compressão de vídeos

Alan Gabriel Godeny

Pedro Henrique de Souza Pozelli

Orientador: Prof. Dr. Rangel Arthur

TCC apresentado à Faculdade de Tecnologia (FT)
como requisito de conclusão do
Curso de Engenharia de Telecomunicações.

Limeira – SP

2020

Agradecimentos do Alan

Aos meus familiares, em primeiro lugar, que me ajudaram e apoiaram, contribuindo para que persistisse com os estudos, mesmos nos momentos mais difíceis.

A minha namorada, por sempre estar do meu lado, me dando conselhos realistas e maduros e me convencendo de que sou maior do que penso.

Ao meu orientador, Rangel Arthur, por possibilitar o presente trabalho e cujas metodologias de ensino considero muito eficazes e estimulantes.

Ao meu amigo, Pedro Pozelli, que ao acaso se tornou minha dupla neste trabalho e foi de imensa ajuda em todos os quesitos.

Agradecimentos do Pedro

A minha família por estar sempre presente, e contribuírem de várias formas para a minha permanência na Unicamp.

Ao Alan Godeny, meu amigo e dupla neste trabalho, por todo seu envolvimento.

Ao meu orientador, Rangel Arthur, por possibilitar e estimular a pesquisa deste tema.

À Unicamp e seus docentes e outros funcionários, muitos dos quais se tornaram achegados, e que tornam possível a minha graduação em uma área formidável.

RESUMO

Um estudo de caráter comparativo foi realizado sobre a compressão e codificação de vídeos, mostrando sua importância e princípios de funcionamento. Três padrões de codificação são abordados e comparados: H.264, H.265, e AV1. São introduzidos conceitos relacionados com a transmissão de vídeos por streaming, em especial a Televisão por IP, bem como métricas de avaliação da qualidade desses serviços.

Palavras-chave: Compressão de vídeos, *codecs*, H.264, H.265, AV1, IPTV.

ABSTRACT

A comparative study was carried out on the compression and encoding of videos, showing their importance and working principles. Three coding standards are addressed and compared: H.264, H.265, and AV1. Concepts related to the transmission of videos by streaming, in particular IP Television, are introduced, as well as metrics for evaluating the quality of these services.

Keywords: Video compression, vídeo encoding, codecs, H.264, H.265, AV1, IPTV.

LISTA DE ABREVIATURAS E SIGLAS

ASF	Sigla do inglês para *Advanced Streaming Format*.
CODEC	Sigla do inglês para *encoder/decoder*.
CTU	Sigla do inglês para *Codification Unit Tree*.
CU	Sigla do inglês para *Codification Unit*.
DASH	Sigla do inglês para *Dynamic Adaptive Streaming over HTTP*.
DCT	Sigla do inglês para *Discrete Cosine Transform*.
DST	Sigla do inglês para *Discrete Sine Transform*.
GOP	Sigla do inglês para *Group of Pictures*.
HLS	Sigla do inglês para *HTTP Live Streaming*.
HTTP	Sigla do inglês para *Hypertext Transfer Protoco*l.
IBGE	Sigla para Instituto Brasileiro de Geografia e Estatística.
IP	Sigla do inglês para *Internet Protocol*.
IPTV	Sigla do inglês para *Internet Protocol Television.*
ITU	Sigla do inglês para *International Telegraph Union.*
LTS	Sigla do inglês para *Long-Term Support*.
ME	Sigla do inglês para *Motion Estimation*.
MOS	Sigla do inglês para *Mean Opinion Score*.
MP4	Padrão de container de áudio e vídeo que é parte da especificação MPEG-4.
MPEG	Sigla do inglês para *Moving Picture Experts Group.*
MPEG-TS	Sigla do inglês para *MPEG transport stream.*
MSS	Sigla do inglês para *Microsoft Smooth Streaming.*
OSI	Sigla do inglês para *Open System Interconnection*.
OTT	Sigla do inglês para *Over The Top*.
PAT	Sigla do inglês para *Program Association Table*.
PES	Sigla do inglês para *Packetized Elementary Stream*.
PMT	Sigla do inglês para *Program Mapping Table*.
PSNR	Sigla do inglês para *Peak Signal-Noise Ratio*.
PU	Sigla do inglês para *Prediction Unit*.
QoE	Sigla do inglês para *Quality of Experience*.
QoS	Sigla do inglês para *Quality of Service*.
QP	Sigla do inglês para *Quantization Parameter*.
RGB	Sistema de codificação de cores.
RTP	Sigla do inglês para *Real-time Transport Protocol.*
RTSP	Sigla do inglês para *Real Time Streaming Protocol*.
SRT	Sigla do inglês para *SubRip Text*.
SSIM	Sigla do inglês para *Structural Similarity Index Measure.*
TCP	Sigla do inglês para *Transmission Control Protocol*.
UDP	Sigla do inglês para *User Datagram Protocol*,

UHD	Sigla do inglês para *Ultra High Definition*.
VMAF	Sigla do inglês para *Video Multimethod Assessment Fusion*.
Y4M	Formato de contêiner.
YCbCr	Sistema de codificação de cores.
YUV	Sistema de codificação de cores.

SUMÁRIO

1 INTRODUÇÃO

A televisão é um veículo de informação e entretenimento ao redor de todo o mundo, sendo o receptor de televisão um dos dispositivos eletroeletrônicos mais populares. Com base em uma pesquisa realizada pelo IBGE, no ano de 2017, 96,7% dos brasileiros possuíam ao menos uma TV em seu domicílio [1].

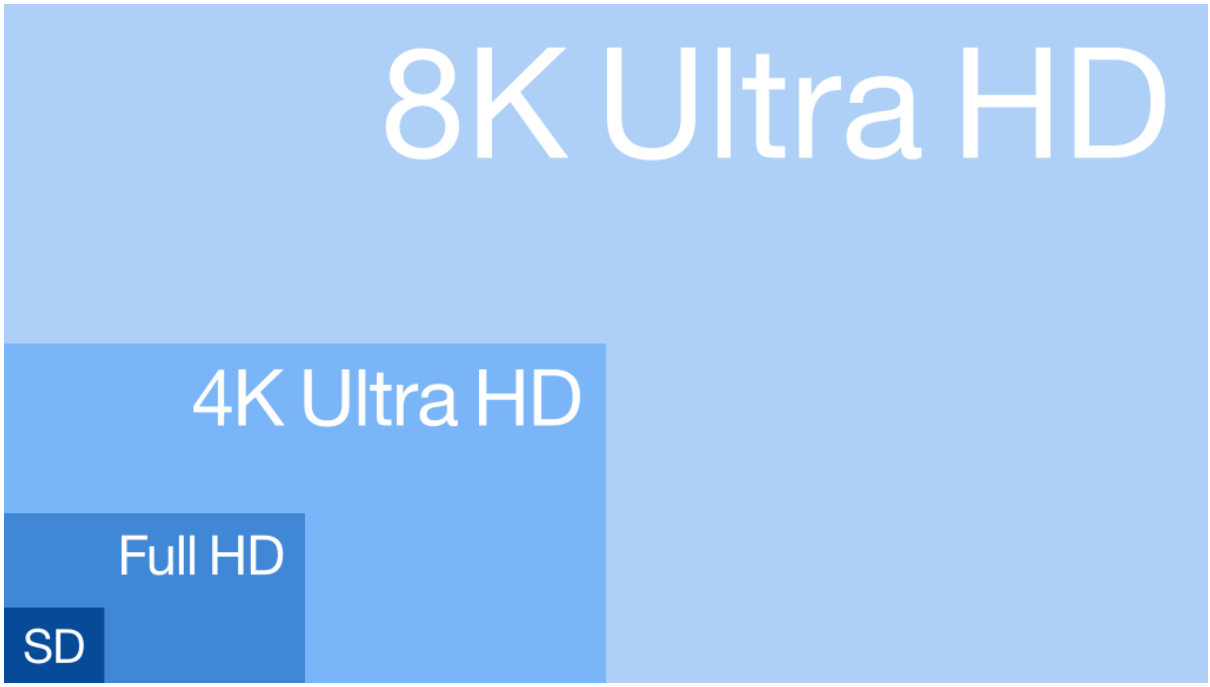
Com a utilização da Internet, o entretenimento em vídeo e a experiência televisiva do usuário ganham novas possibilidades. A capacidade de transmissão de dados possibilita a interatividade em diferentes níveis, traz potenciais benefícios e facilidades atualmente disponíveis apenas na Internet [2]. Alguns exemplos dessas possibilidades são a educação a distância, o entretenimento digital, o acesso a notícias e serviços [3].

A distribuição de conteúdo em vídeo por meio da Internet tem aumentado no Brasil [4]. As plataformas de conteúdo têm se tornado mais acessíveis; segundo uma projeção, o crescimento anual será de 16,7% até 2025 [5].

A plataforma de acesso a conteúdo televisivo por meio da Internet é conhecida como IPTV (do inglês *Internet Protocol Television*). Os serviços de IPTV são entregues por operadoras, também conhecidas como provedores de serviço, que utilizam o protocolo IP (do inglês *Internet Protocol*) de ponta a ponta. Na ponta do usuário, um receptor de IPTV, também conhecido como *set-up box*, concede acesso aos canais televisivos [6].

As operadoras precisam acompanhar o passo do desenvolvimento dos novos formatos e resoluções de TV. Por exemplo, dois formatos populares são a TV de definição padrão (ou SDTV) e a TV de alta definição. Já estão aparecendo novos formatos no mercado, como a televisão de ultra alta definição (UHDTV), com resoluções de 4K e 8K. A proporção entre esses formatos é mostrada na *Figura 1*.

Figura 1 - Comparativo de resoluções



Fonte: Wikimedia Commons. Disponível em: https://commons.wikimedia.org/wiki/File:Resolution_of_SD,_Full_HD,_4K_Ultra_HD_%26_8K_Ultra_HD.svg. Acesso em 17 de junho de 2019.

À medida que a definição da imagem aumenta, mais se exige da rede [7]. Diante da necessidade de cada vez maiores taxas de bits para a transmissão de vídeo de alta qualidade, assegurar a qualidade do serviço e a qualidade da experiência do usuário constituem desafios para as operadoras [6]. Assim, a busca por padrões de compressão eficientes é acelerada [8].

1.1 Visão geral do trabalho

O presente trabalho visa apresentar o funcionamento de alguns dos atuais padrões de compressão e codificação de vídeo, especificamente os padrões H.264, H.265, e AV1. Serão introduzidos conceitos da compressão de vídeos e será apresentado o impacto da evolução das técnicas utilizadas, tendo como referencial a sua importância nos serviços de televisão pela Internet. Como será demonstrado, cada padrão de compressão apresenta vantagens e desvantagens, e nem sempre um padrão mais atual é o mais utilizado.

Neste capítulo, será feita uma introdução à televisão pela Internet, e uma distinção entre os serviços de IPTV e da similar OTT (do inglês *Over-the-top*). Também

serão apresentados parâmetros utilizados para análise da qualidade desses tipos de serviço.

No *Capítulo 2*, serão apresentados conceitos sobre a formação de vídeos e a transmissão de vídeos pela Internet. Também será feita uma introdução aos princípios de compressão e codificação de vídeos, e como estes podem ser transportados pela rede. Em seguida, no *Capítulo 3*, serão introduzidos padrões de codificação e seus princípios gerais de funcionamento.

O *Capítulo 4* apresenta a metodologia, os resultados obtidos no escopo deste trabalho, discussões comparativas sobre os parâmetros de qualidade calculados, e finalmente uma discussão sobre os resultados obtidos.

1.2 Televisão pela Internet: IPTV e OTT

Conectar a televisão com a Internet torna possível uma experiência muito interativa, em que informação pode também ser transmitida do assinante para o provedor. Algumas das possibilidades são as enquetes, votações, shows interativos, e cenas escolhidas por cada assinante, aumentando a possibilidade de novas combinações.

Outro recurso interessante é a possibilidade de reprodução de conteúdo gravado, em que o assinante pode visualizar programações anteriormente exibidas retrocedendo a linha do tempo de exibição.

A distribuição de televisão ao vivo através da Internet pode ser realizada com a utilização de duas abordagens, que são os serviços de IPTV e os serviços de OTT.

Como existe certa similaridade das tecnologias empregadas em ambos os tipos de serviço, será feita uma distinção a seguir com as suas características comuns e exclusivas.

As duas abordagens são semelhantes porque a distribuição de conteúdo em ambas é realizada através de protocolos do tipo IP. A principal distinção é que, enquanto que o IPTV utiliza de uma rede destinada para esse fim, os serviços de OTT fazem uso das estruturas de redes já existentes (da Internet pública) para oferecer conteúdo diretamente aos seus utilizadores [9].

No caso dos serviços de IPTV, os provedores de conteúdo incluem redes de televisão existentes, provedores de conteúdo e empresas independentes de provedores de infraestrutura.

Em um serviço de IPTV, os elementos principais são: o “*head end”*, em que os conteúdos dos canais são formatados para distribuição ao vivo; a “rede principal”, que transporta os dados pela rede do provedor; a “rede de acesso”, conectada à rede principal, que faz a conexão de banda larga do provedor à rede doméstica do usuário final, que é conectada aos aparelhos de televisão via *set-top box [6]*.

Como a IPTV se utiliza de uma estrutura própria para distribuição de conteúdo, a sua disponibilidade é afetada pela limitação nas áreas de cobertura onde a empresa provedora atua. Em contraste, fica claro que o acesso à tecnologia dos serviços de OTT se faz disponível em qualquer localidade que possua uma conexão com a rede de Internet.

Os custos de implementação para os serviços de IPTV são inicialmente mais elevados, uma vez que demandam a instalação de estruturas próprias limitadas para cada localidade. Nos serviços de OTT, por não haver dependência de uma estrutura única, é possível atender utilizadores de diferentes localidades, visto que não é necessário que o provedor possua infraestrutura instalada nas proximidades de cada usuário.

Em relação à qualidade do serviço e da experiência, nos serviços de IPTV é possível garantir uma largura de banda exclusivamente para a aplicação, visto que o provedor possui conhecimento acerca das infraestruturas disponíveis para utilização, podendo garantir ao usuário alta qualidade de exibição.

Nos serviços de OTT, a garantia da estabilidade e da qualidade para o utilizador depende da estrutura de rede pela qual o sinal digital irá trafegar, podendo haver perdas parciais ou até completas de trechos do conteúdo transmitido. Nesse caso, ocorrem congelamentos nas imagens ou perdas de detalhamento na definição da imagem [10].

1.3 Parâmetros de medição de qualidade

Os provedores de serviços de IPTV buscam manter a qualidade de seus serviços a partir de duas métricas principais: a qualidade da experiência, ou QoE (do inglês *Quality of Experience*), e a qualidade do serviço, ou *QoS* (do inglês *Quality of Service*). Em geral, o QoS é determinado pelo desempenho da rede [11].

A qualidade da experiência indica o grau de satisfação diante das expectativas que os usuários têm ao assistirem ao serviço de vídeo oferecido. Para que o serviço seja competitivo, a qualidade experimentada pelos assinantes da IPTV deve ser pelo menos igual aos outros serviços de TV fornecidos via cabo ou satélite [12].

A QoE pode ser analisada por parâmetros subjetivos e objetivos, que são correlacionados [13]. Os parâmetros subjetivos geralmente são mais significativos, porque alguns efeitos psicovisuais podem não ser bem representados por valores numéricos. Os parâmetros objetivos, como o PSNR, podem ser repetidos mais extensivamente por serem calculados computacionalmente [13, 14].

Os parâmetros subjetivos fornecem um modelo para que a qualidade visual do vídeo seja analisada. Para isso, um grupo de pessoas é montado e cada pessoa avalia em uma escala a sua respectiva experiência de percepção da qualidade do vídeo.

O parâmetro subjetivo mais utilizado é o MOS (do inglês *Mean Opinion Score*), que representa uma pontuação média baseada na opinião de um grupo de pessoas de teste, não sendo comum a atribuição de uma pontuação ao vídeo inicial [14, 15].

Os parâmetros objetivos de livre utilização mais populares são: SSIM, VMAF, PSNR. Esses parâmetros são calculados por comparação do vídeo inicial (antes da codificação) com o vídeo final (após a codificação), quadro por quadro, e daí pode se obter uma média geral [14]. O SSIM e o PSNR são parâmetros com base matemática, sendo que, dentre os quatro parâmetros, o SSIM é o menos utilizado, e o PSNR é o mais utilizado, embora este tenha a menor eficiência [16, 17].

O PSNR calcula a relação sinal-ruído de pico entre a imagem original e a imagem comprimida. Quanto maior for seu valor, melhor é a qualidade da imagem comprimida. O SSIM é uma métrica que assume que a degradação na imagem se

relaciona com a mudança em informação estrutural, por exemplo, nas formas e bordas. Comparado ao PSNR, o SSIM compara imagens de forma mais semelhante à do sistema visual humano [17].

Para calcular o PSNR, é necessário calcular o erro quadrático médio (MSE), que representa a diferença acumulada entre as imagens, e a máxima flutuação possível na imagem (MAX_I), que é um valor constante baseado na quantidade bits usados para cada pixel na representação da imagem. A seguinte equação é então utilizada para calcular o PSNR:

$$PSNR = 10\ log \frac{MAX_I^2}{MSE}\ [\mathrm{dB}] \qquad (1)$$

Em relação ao SSIM, um vídeo codificado recebe pontuação de 0 a 1 (melhor); em relação ao PSNR, a escala é de 0 a 100 (melhor). Vale destacar que na determinação dos parâmetros PSNR e VMAF, é essencial que os vídeos pré-codificação e pós-codificação tenham a mesma resolução [17].

O modelo do parâmetro VMAF é de código aberto, o que contribui para sua ampla utilização. Na prática, este realiza uma aproximação da qualidade de percepção, e é conhecido por ser bastante eficiente, utilizando de técnicas modernas de aprendizado de máquina [18]. Os algoritmos para o cálculo de VMAF foram desenvolvidos pela *Netflix*.

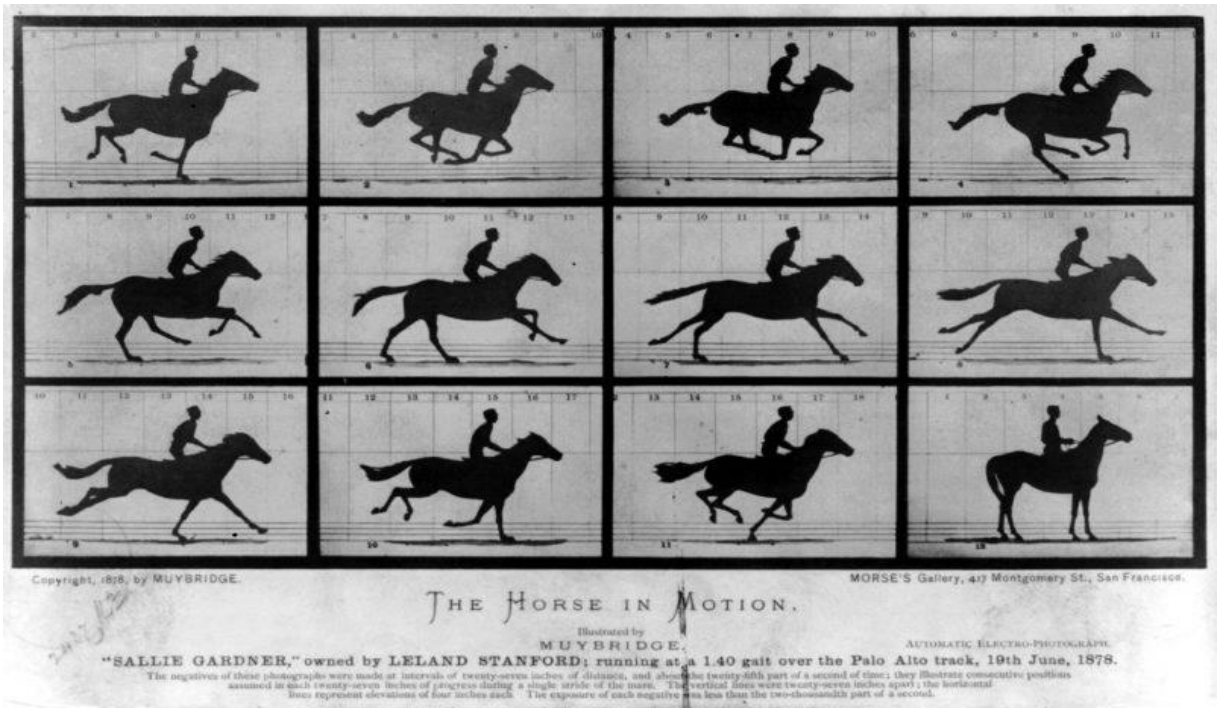
2 COMPRESSÃO

Como já comentado, um vídeo é formado por imagens digitais chamadas quadros. Uma imagem digital é a representação de uma imagem em duas dimensões por meio de um conjunto finito de elementos digitais, que em uma tela são pontos luminosos chamados píxeis [19].

Matematicamente, a imagem digital é representada por uma função $\mathrm{f}(\mathrm{x}, \mathrm{y})$ de duas variáveis, x e y, que formam um par ordenado no plano cartesiano para cada um dos píxeis. Por exemplo, numa imagem em escala de cinza, o valor de cada ponto é um valor inteiro proporcional ao brilho da imagem na respectiva coordenada [20].

No escopo deste trabalho, um vídeo é considerado como um conjunto de imagens estáticas dispostas em sequência, em que cada uma dessas imagens é chamada quadro. Como mostra a *Figura 2*, a exibição consecutiva de quadros cria a sensação de movimento dos objetos neles contidos, e a taxa na qual isso é realizado influencia a percepção de continuidade desse movimento [20]. Com isso em vista, a trilha sonora dos conteúdos em vídeo não será considerada.

Figura 2 - Sequência de quadros e a sensação de movimento



Fonte: The Vintage News. Disponível em https://www.thevintagenews.com/wp-content/uploads/2016/01/The_Horse_in_Motion.jpg. Acesso em 20 de junho de 2020.

2.1 Introdução à compressão de vídeos

Quanto maior é o nível de detalhamento de uma imagem, maior é a quantidade de píxeis utilizada para representá-la, e consequentemente maior é a quantidade de informações digitais que devem ser armazenadas [21]. Assim, por serem formados a partir de imagens sequenciais, fica claro que a quantidade de informação necessária para representar um vídeo está diretamente relacionada com a qualidade visual de seus quadros.

Como os meios físicos de armazenamento, processamento e transmissão de dados possuem limites, a quantidade de informação necessária para representar um arquivo de vídeo é de grande relevância [22].

No contexto de armazenamento digital, uma unidade de medida comum é o *byte*, que é formado por um grupo de oito *bits*. Como cada *bit* pode assumir dois valores, um *byte* pode representar 256 valores. O espaço necessário para o armazenamento de uma imagem pode ser medido em *bytes*. Por exemplo, no sistema RGB, um píxel é representado por três *bytes*, isto é, 256 níveis para cada uma das suas três cores componentes.

Uma métrica utilizada para mensurar o fluxo de dados de um vídeo é a taxa de *bits*, que consiste na quantidade de *bits* por segundo necessários para a representação do vídeo. Quanto maior a taxa de *bits*, maior o detalhamento das imagens dos quadros transmitidos em um intervalo de tempo e, consequentemente, maior é a qualidade do vídeo e também o seu tamanho.

Por exemplo, considere que um vídeo seja formado por 30 quadros por segundo, e que cada quadro possua 1.920 colunas por 1.080 linhas de píxeis, ou seja, 2.073.600 píxeis, e que para a exibição em cores cada píxel seja representado por três *bytes* (24 *bits*). A cada segundo do vídeo, é necessário 1,49 *Gbits* (bilhões de *bits*) para representá-lo. Neste caso, a taxa de *bits* para sua transmissão é 1,49 *Gbits/s*.

Para um melhor aproveitamento dos meios físicos de transmissão, tendo em conta seus limites, são utilizadas técnicas de compressão para reduzir a quantidade de *bits* transmitidos para uma mesma quantidade de informação [23].

A aplicação de técnicas de compressão de vídeo possibilita a economia de dados, maximizando a disponibilidade das redes de telecomunicações, visto que menos *bits* serão trafegados por meio dessas estruturas, que são em grande parte ocupadas por dados de vídeos [24]. Com a crescente demanda por vídeos de alta definição, a capacidade de tráfego na rede exigida é cada vez maior, sendo necessário que formas mais eficientes de compressão de vídeos estejam sempre sendo aprimoradas.

As técnicas de compressão podem ser com perdas ou sem perdas. No tipo sem perdas, também chamado compactação, os dados reconstruídos no decodificador são uma cópia perfeita dos dados originais, como se não houvesse ocorrido a codificação. A compressão sem perdas impede o acúmulo de ruído na imagem, pois os efeitos adversos da compactação são multiplicados a cada repetição do processo [25].

Na compressão com perdas, ocorre o descarte de parte da informação considerada redundante, de modo que sua recuperação não pode ser completa, tal como acontece na compressão sem perdas. Por isso, podem ser atingidas altas taxas de compressão em troca de alguma perda de qualidade visual [26].

2.2 Introdução aos codecs

No processo de transmissão de vídeo, é comum a utilização de um *codec* (abreviação do inglês *encoder/decoder*). Um *codec* é formado por um codificador e um decodificador [25].

Os *codecs* têm como objetivo reduzir a taxa de transmissão de *bits*. Esse processo é usado para viabilizar a transmissão de grandes volumes de dados pela Internet. Por exemplo, em transmissões de sinal de televisão, a utilização dos *codecs* de vídeo pode reduzir a taxa de transmissão de *bits* em até mais de 100 vezes [22].

Devem assim representar os dados do vídeo usando o menor número de *bits* possível e com uma fidelidade tão alta quanto possível. É um desafio satisfazer ambos os requisitos simultaneamente, uma vez que um menor número de *bits* tipicamente produz uma imagem de qualidade reduzida.

O tamanho do vídeo comprimido é controlado pela taxa de compressão de acordo com a qualidade desejada. Por exemplo, para uma transmissão esportiva, como uma partida de futebol, a preservação da qualidade dos movimentos é muito importante. Já na transmissão de um leilão de joias, a qualidade das cores ou texturas das imagens é a parte mais importante.

Cada quadro de um vídeo pode conter muitos detalhes que não carregam informação importante e cuja eliminação não irá afetar a qualidade do vídeo. A diminuição da taxa de *bits* durante a compressão é alcançada pela remoção de redundância presente nos dados, ou seja, a remoção de componentes repetitivos e que são dispensáveis na reprodução fiel dos dados [25]. A seguir, serão analisadas algumas formas de redundância.

2.3 Redundância Espacial

A redundância espacial ocorre quando píxeis vizinhos num quadro estão correlacionados, como acontece, por exemplo, num objeto de uma única cor. A correlação entre os píxeis de uma imagem é a medida de quanta informação os píxeis têm uns sobre os outros, e o aproveitamento dessa correlação permite obter por aproximação a cor e o brilho de píxeis adjacentes em cada quadro de um vídeo [26].

Por exemplo, na *Figura 3*, em um quadro onde um avião passa no céu com poucas nuvens, a informação relevante é o avião, e o fundo azul é praticamente uniforme.

Figura 3 - Aproveitamento da similaridade entre píxeis

Fonte: Flickr. Disponível em https://www.flickr.com/photos/josh_marvin/6261285432/. Acesso em 16 de maio de 2020.

Na prática, para distinguir dados importantes dos não importantes, podem ser utilizadas a redundância de píxeis adjacentes e as características de reduzida sensibilidade do olho humano às altas frequências espaciais [25]. Com esse objetivo, é realizada uma transformada chamada DCT (do inglês *Discrete Cosine Transform*), que por si própria serve como uma forma de compressão sem perdas e é um processo completamente reversível [26].

Para que haja maior compressão, são introduzidas as perdas. Em um processo denominado segmentação de imagem (ou *thresholding*, do inglês*)*, blocos de um quadro recebem a aplicação da DCT. Os coeficientes obtidos são então quantizados com uma precisão inversamente proporcional às frequências espaciais. Assim, os dados de píxeis com alta frequência que contenham baixo conteúdo de energia podem ser descartados, sem causar degradações perceptíveis na imagem [27].

Dessa forma, a aplicação da DCT introduz perdas, pois parte das informações foi completamente descartada e não pode mais ser recuperada, reduzindo consideravelmente a taxa da informação a ser transmitida. A qualidade da imagem codificada relaciona-se diretamente com a eliminação dessas altas frequências, e então pode ser obtido um compromisso entre a qualidade da imagem e a taxa de compressão, considerando-se as especificações do meio de transmissão disponível [27].

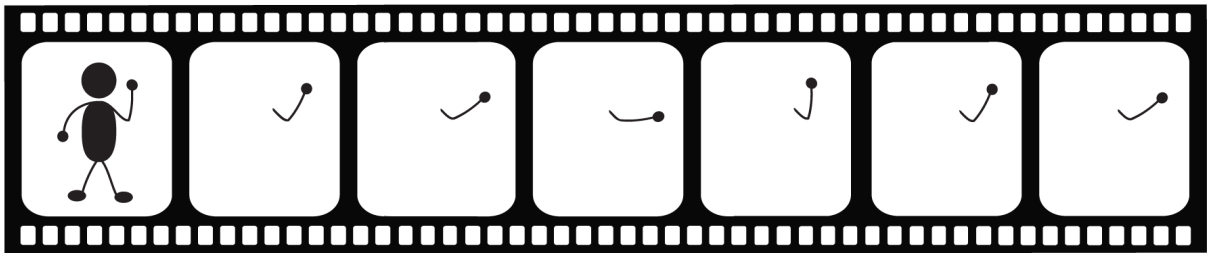
2.4 Redundância Temporal

A redundância temporal ocorre quando partes dos dados não mudam entre quadros consecutivos. É possível aproveitar a similaridade entre quadros sucessivos que formam algum tipo de movimento em uma cena. Como o intervalo entre dois quadros é bem pequeno, os quadros consecutivos são muito similares, de onde parte a ideia de redundância temporal [28].

A ideia é expressar as diferenças entre os quadros por se levar em conta o deslocamento dos objetos, em vez de transmitir os quadros por inteiro. Para isso, um quadro é dividido em blocos de tamanho variável. Em cada bloco, a detecção do movimento é realizada por se encontrar no quadro seguinte a melhor correspondência para o bloco na vizinhança da posição original deste [28].

Um exemplo prático está na *Figura 4*, em que o personagem está movimentando apenas o braço. Em vez de se codificarem os dados completos dos quadros sucessivos, é possível codificar apenas a informação do quadro inicial juntamente com as mudanças da posição do braço.

Figura 4 – Representação da diferença entre quadros



Fonte: Adaptado de B&H Photo Video. Disponível em https://static.bhphotovideo.com/explora/sites/default/files/02intraframes-interframes.png. Acesso em 05 de junho de 2020.

Nos casos em que muitas informações novas precisam ser codificadas em quase todos os quadros, é mais difícil codificar parte do conteúdo do vídeo. Por exemplo, os carros de corrida se movem e giram com alta velocidade entre quadros próximos.

2.5 Redundância perceptiva

A redução de redundância perceptiva considera as capacidades do sistema visual humano, que não trata toda a informação visual com a mesma relevância; na realidade, é mais sensível (percebe melhor) à intensidade luminosa (brilho) que à cor. Esse fato possibilita que a representação dos quadros seja realizada com menos informação, isto é, que os componentes de cor sejam representados com menor definição que os de luminância [29].

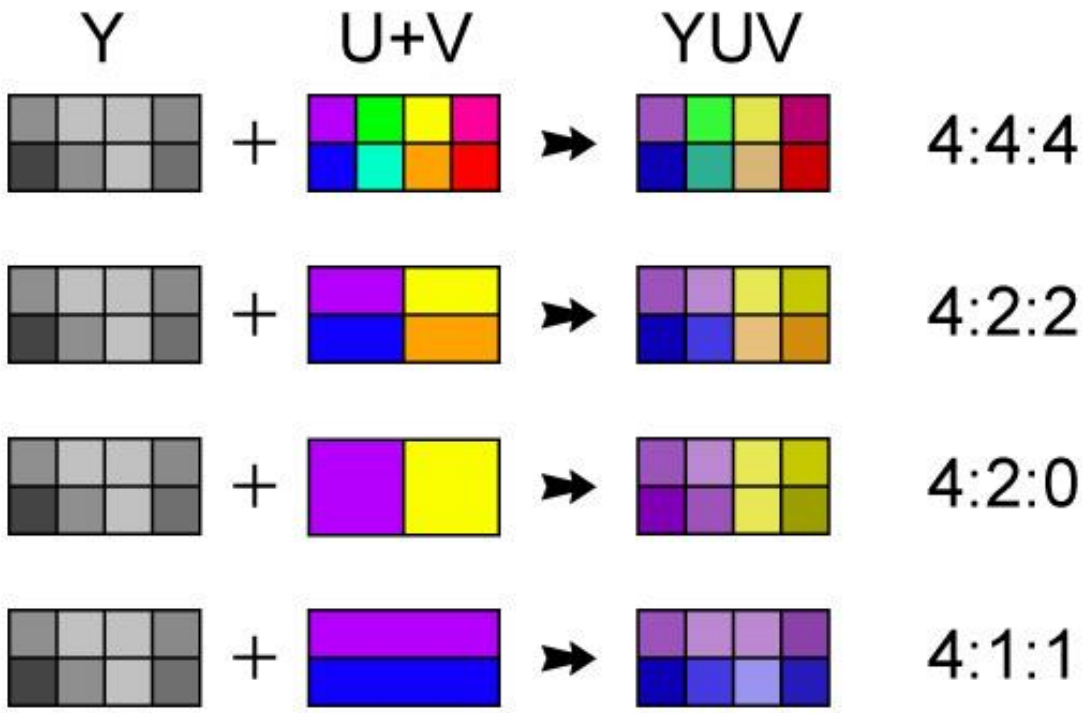
Existem diversos modelos de cores, por exemplo, RGB, YUV, CIE e CMYK. Principalmente por sua capacidade de representar separadamente a intensidade luminosa e a cor e por sua compatibilidade com o padrão YUV usado nos sistemas de televisão, o modelo YCbCr é usado em praticamente todos os padrões de codificação de vídeo.

Basicamente, um quadro é dividido em macroblocos que são então divididos em três camadas. Cada camada contém uma componente do modelo de cor YCbCr. Para a componente de luma (Y) são usados blocos de 16x16 píxeis ou 4 blocos de 8x8 píxeis. Para os componentes cromados com diferença de azul (Cb) e os cromados com diferença de vermelho (Cr), blocos menores (por exemplo, 8x8 píxeis) podem ser utilizados para representar blocos maiores, o que é conhecido como subamostragem.

A *Figura 5* ilustra um modo em que não há subamostragem (modo 4:4:4), isto é, em que não há redução da redundância perceptiva. Também mostra três modos em que ocorre subamostragem (modos 4:2:2, 4:2:0 e 4:1:1). No modo 4:2:2, metade da informação é descartada pois, para as componentes de cor, um bloco passa a representar dois.

Nos modos 4:2:0 e 4:1:1, um bloco passa a representar quatro blocos; consequente, a informação transmitida é quatro vezes menor, e sua diferença se dá apenas entre as resoluções horizontal e vertical. A diferença de cores em uma situação real, para píxeis adjacentes, não costuma ser tão brusca como no exemplo.

Figura 5 - Subamostragem de componentes no modelo de cores YCbCr



Fonte: Rave Publications. Disponível em https://www.ravepubs.com/chroma-subsampling/. Acesso em 15 de março de 2020.

2.6 Efeitos da compressão e da codificação

A remoção de dados na codificação do vídeo causa perda da nitidez das imagens. Essa perda dificulta, por exemplo, a leitura de textos no vídeo.

Distorções na imagem causadas na reconstrução da imagem depois da descompressão causam o efeito de serrilhado (ou *aliasing*), que pode se apresentar em várias formas. Duas de suas formas comuns são o efeito de escada e o efeito estroboscópico.

O efeito de escada consiste em uma aparência irregular dos contornos, exibidos de maneira brusca, dando uma aparência de escada (visto mais facilmente com a imagem ampliada). O efeito estroboscópico é aquele que, em filmes, causa a impressão de que uma roda parece girar em velocidade diferente da real ou mesmo no sentido contrário, e tem como causa uma incompatibilidade entre a velocidade de rotação e a taxa de quadros da compressão.

Quanto maior for a qualidade do vídeo codificado, ou fidelidade em relação ao original, maior será o seu tamanho, havendo um compromisso entre as duas partes, e a escolha de qual parâmetro será mais prejudicado depende, como já visto, de fatores como o tipo de conteúdo e da infraestrutura de transmissão.

Nesse sentido, dois fatores afetam o tamanho e a qualidade final do vídeo codificado: o conteúdo a ser preservado do vídeo original e a configuração do *codec* utilizado. Entre essas configurações, estão a taxa de quadros e a resolução. Por exemplo, a diminuição da taxa de quadros do vídeo pela eliminação de quadros reduz o tamanho do arquivo final.

2.7 Padrões para codecs

Padrões de codificação são criados para possibilitar que produtos de diferentes fabricantes possam ter compatibilidade. A escolha de um padrão apropriado é muito importante para os produtores de conteúdo para transmissão. Estes precisam conhecer, dentre outros, as propriedades, vantagens, desvantagens, e custos que terão em cada em cada escolha.

Embora neste trabalho sejam abordados apenas os padrões relacionados com vídeo, é interessante pontuar que existem padrões para codificação de áudio, que estão fora do escopo deste trabalho, mas que geralmente são usados em conjunto com os padrões de codificação de vídeo.

Um codec de vídeo é basicamente formado por um codificador e um decodificador. Um codificador comprime o vídeo, formando um fluxo de vídeo, enquanto o decodificador decodifica o fluxo de vídeo. Os padrões para *codecs* definem o formato de um vídeo comprimido e um método para decodificá-lo, mas não padronizam a codificação. O *codec* é a implementação de uma especificação, que por sua vez é o formato de codificação.

Como será reforçado mais adiante, o melhor desempenho da compressão tem maior custo computacional tanto na codificação como na decodificação.

2.8 Fluxo de transporte

Depois que o vídeo é codificado para a transmissão, o próximo passo é transportá-lo pela rede. A seguir, será feita uma visão geral do fluxo transporte para o formato MPEG-TS (sigla do inglês que significa fluxo de transporte MPEG). Os conceitos são reaproveitados para os formatos H.264, H.265, e AV1.

Inicialmente, os fluxos elementares que chegam do codificador recebem um cabeçalho, que inclui metadados para identificação e sincronia, formando os pacotes chamados PES. Esses pacotes são fracionados, e cada um recebe um cabeçalho, passando a conter ao todo 188 *bytes*. Esses pacotes em sequência constituem o fluxo de transporte, que pode conter dados de vídeo, áudio e dados.

Entre outros, a identificação dos pacotes possibilita que se intercalem programas de TV em um mesmo fluxo de transporte. Nesse caso, para que se possa filtrar os pacotes do programa que está sendo exibido, existem dois tipos de pacotes auxiliares, que são a Tabela de Associação de Programa (PAT) e a Tabela de Mapa do Programa (PMT). No caso dos serviços de IPTV, para que a taxa de *bits* não seja muito alta, apenas o programa que será apresentado é transmitido.

2.8.1 Contêineres de mídia

Os protocolos de rede são organizados em estruturas chamadas pilhas, baseada no esquema de camadas *Open Standards Interconnect* (OSI), com o nível mais baixo sendo a camada física e o mais alto a camada de aplicação, onde os protocolos de *streaming* se encontram. Nesse esquema de camadas, os dados são agrupados em contêineres, que vão ficando mais estruturados à medida que atravessam cada camada.

O contêiner é o arquivo que encapsula o vídeo, as trilhas de áudio, as legendas, e outros dados. Exemplos de formatos de contêiner atuais são MP4, Matroska, MPEG2-TS e ASF.

Existem diversos formatos de contêineres; cada um tem suas respectivas compatibilidades, aplicações, e conjunto de *codecs* de vídeo e áudio que podem transportar. Assim, a escolha de um contêiner depende dos requisitos do projeto, como: o suporte dos dispositivos e software ao formato, e a disponibilidade de obtenção das licenças.

Por exemplo, alguns contêineres suportam apenas áudio, enquanto outros suportam áudio e vídeo. Além disso, os contêineres geralmente podem conter fluxos gerados por vários tipos de *codecs*. Para obter maior compatibilidade, como as plataformas de reprodução aceitam geralmente alguns tipos de *codecs*, convém disponibilizar a transmissão em diferentes formatos de contêineres com diferentes opções de *codecs*.

O formato MP4 é a versão mais recente do formato de arquivo MPEG, atualmente descrito na especificação MPEG-4 Part 14 [30]. É um contêiner popular por suportar vários dos *codecs* mais utilizados e ser amplamente suportado [31].

O encapsulamento dos dados do vídeo ocorre como descrito a seguir. Após a geração do conteúdo, os fluxos codificados de dados e metadados são encapsulados em contêineres, que são transportados pela rede até alcançar o dispositivo do usuário final, onde são abertos (no processo inverso ao encapsulamento), e o conteúdo é decodificado e finalmente reproduzido.

Os metadados servem para levar ao dispositivo de reprodução informações sobre o vídeo, por exemplo, tempo de reprodução, duração e os formatos de codificação utilizados. Assim, podem ser realizadas a decodificação e a reprodução dos trechos de mídia contidos no vídeo de forma sincronizada.

2.8.2 Protocolos de transmissão

Os contêineres podem ser transmitidos ao vivo pela rede por meio dos protocolos TCP, UDP, SRT, HTTP (MPEG-DASH, HLS). Existe também a possibilidade de o vídeo ser transferido sem um contêiner, como faz o protocolo WebRTC, que transfere vídeo e áudio em conexões separadas, utilizando o protocolo RTP [42].

O protocolo universal de datagrama (ou UDP, do inglês *User Datagram Protocol*) busca a entrega pontual de um pacote de um nó para o próximo, mas não garante a entrega de uma série completa de pacotes, nem que todos cheguem na ordem correta. O protocolo de controle de transmissão (ou TCP, do inglês *Transmission Control Protocol*) elimina as deficiências do UDP, garantindo que todos os pacotes serão enviados e entregues ao destinatário na sequência correta.

Na transmissão via TCP, um pacote atrasado atrasa a reprodução; na transmissão via UDP, um pacote atrasado fica fora de sequência e não pode mais ser utilizado, o que causa queda da taxa de quadros e prejudica a experiência de visualização.

O Protocolo de *Streaming* em Tempo Real (RTSP) pode usar os protocolos na camada de transporte (UDP e TCP) para transmitir dados. O Protocolo de Transporte em Tempo Real (RTP) usa apenas o UDP para transporte de dados. Atualmente protocolos baseados em HTTP, como o *Microsoft Smooth Streaming* (MSS) e o *HTTP Live Streaming* (HLS) da *Apple* são amplamente utilizados.

2.8.3 Problemas na transmissão

Os problemas na transmissão, como a perda de pacotes, são um grande desafio na garantia da experiência de usuário [33]. Em diversos tipos de transmissão, é feito o uso de redundância na codificação para possibilitar que dados ausentes sejam reconstruídos, o que, por outro lado, aumenta a carga no processamento [34].

Um codec apresenta estratégias para lidar com partes ausentes da imagem quando ocorrem perdas de dados. O formato H.264 especifica que o decodificador deve guardar em *cache* uma grande quantidade de quadros, para que assim os possa utilizar quando houver perda de dados [35]. Desta forma, um quadro reconstruído pode usar trechos de vários quadros diferentes.

2.8.4 Aspectos da codificação para IPTV

A ideia básica da transmissão é que um servidor, por meio de uma conexão de rede, fornece aos receptores de TV um fluxo de *bits* que é decodificado, de forma que os quadros do vídeo possam ser reconstruídos.

Esse fluxo de dados deve ser precisamente sincronizado e sem interrupções, e a quantidade de dados deve ser suficiente para que a experiência

audiovisual seja agradável. Por se basearem na transmissão pela rede, a qualidade dos serviços de IPTV fica vulnerável a problemas que ocorram na rede.

Quando se comparam a codificação para gravação e a codificação para transmissão ao vivo, a principal diferença é que atrasos na codificação para transmissão ao vivo levam a problemas na reprodução contínua.

Na transmissão ao vivo, a codificação precisa ser realizada em tempo real, com rapidez suficiente para que o fluxo de saída acompanhe a reprodução do vídeo [46]. Além disso, os níveis de tráfego podem comprometer a disponibilidade de largura de banda em um servidor ou na recepção, o que sustenta a busca por altas taxa de compressão, visando a diminuição da taxa de *bits* [47].

Para atender a essas necessidades, o processo de codificação e *streaming* requer poder computacional elevado, o que aumenta os custos com *hardware*. Por exemplo, para prover alta qualidade em baixas taxas de *bits* se requer mais do codificador, o que aumenta o tempo de processamento. Para diminuir esse tempo, torna-se necessário hardware mais robusto, o que aumenta os custos.

Por outro lado, uma vez que exigir mais do hardware para decodificação reduz o público-alvo, é desejável manter baixa (tanto quanto possível) a complexidade do processo de decodificação [36]. Por isso, o equilíbrio, ou compromisso, entre a taxa de compressão e a complexidade de decodificação se torna uma necessidade.

Assim, de forma geral, são de interesse a baixa taxa de *bits*, a maior qualidade audiovisual do conteúdo, a maior rapidez e o menor custo de hardware para os processos de codificação e decodificação. A partir disso se nota a importância da eficiência dos *codecs*, e a necessidade de ceder em um ou mais desses itens em troca de alguns dos outros.

Os serviços de IPTV, na tentativa de aproveitar ao máximo a capacidade de rede disponível, recorrem a algumas técnicas que serão comentadas a seguir.

- *Buffers*: Os pacotes de rede são suscetíveis a atrasos e a perderem sua sequência; por isso, os dispositivos de reprodução de vídeo armazenam certa porção de conteúdo antes de iniciar a reprodução (criando um *buffer*), propiciando uma chance razoável de evitar congelamentos na imagem ou no som, uma vez que esses congelamentos seriam necessários para carregar mais conteúdo. A necessidade de

buffers, ou sua quantidade, é maior, por exemplo, em conexões em que a taxa de *bits* é baixa ou quando o conteúdo apresenta baixa taxa de compressão.

- Taxa de *bits* adaptável: Um *player* pode solicitar um fluxo a uma taxa de *bits* baixa, com qualidade inferior, quando falhas ou congestionamentos na rede comprometem a reprodução contínua do conteúdo. Da mesma forma, pode utilizar de uma taxa de *bits* maior se as condições de rede o permitirem. O servidor se beneficia por evitar desperdício de largura de banda [37].

3 PADRÕES DE CODIFICAÇÃO DE VÍDEO

O padrão H.264 foi criado em 2003 pela *Joint Video Team* nas especificações ITU H.264 e MPEG-4 Part 10. Com o passar dos anos, passou a ser amplamente utilizado para a codificação na transmissão de vídeos por grandes distribuidoras de conteúdo, suportando resoluções de até 8192x4320 junto com 8K UHD. O *YouTube* implementou sua utilização em dezembro de 2008.

Na busca por trazer melhor eficiência na compressão, o H.264 possibilita um compromisso entre maior qualidade de imagem e menor taxa de *bits* ou vice-versa, trazendo maior flexibilidade nos processos de compressão, transmissão e armazenamento do vídeo [38].

O fluxo codificado no formato H.264 pode ser encapsulado em diversos contêineres, sendo muito comum encontrá-lo no formato MPEG-4 e MPEG-TS.

O padrão H.265, especificado na MPEG-H Part 2, também foi desenvolvido pela *Joint Video Team*, com o objetivo de gerar arquivos com menor utilização da taxa de *bits* e com qualidade de imagem similar aos do seu antecessor H.264. Foi elaborado para atender à codificação e à decodificação de vídeo em resoluções muito altas, incluindo 8K [39]. A transmissão em 4K é viabilizada com o H.265, porque as taxas de bits são consideravelmente menores.

O padrão AV1 foi desenvolvido pela *Alliance for Open Media*. Seu desenvolvimento busca a codificação de vídeo para entrega em tempo real e de alta qualidade, a escalabilidade de dispositivos, a diversidade de taxas de *bits*, e a flexibilidade para conteúdo tanto comercial como gratuito [40].

O AV1 ainda está na fase experimental, por isso sua compatibilidade não é tão alta como a dos outros padrões, e sua codificação e decodificação ainda demandam tempo consideravelmente grande.

3.1 Funcionamento dos codecs

3.1.1 Perfis

Visto que diferentes perfis estão disponíveis no formato de codificação, há diferentes algoritmos para comprimir o vídeo e criar o fluxo codificado, que visam diferentes plataformas ou tipos de dispositivo.

Por exemplo, alguns perfis favorecem dispositivos com *hardware* mais simples, em troca de certa perda na qualidade visual. Para serviços de TV baseados em IP, um perfil com menor complexidade de codificação tende a ser desejável, pois o tempo de codificação é importantíssimo nesse cenário [41].

Por exemplo, existem vários perfis para o H.264, cujas características podem ser classificadas em quatro tipos principais. A maioria dos perfis usa componentes de cores de 8 *bits* e subamostragem de cor de 4:2:0.

O primeiro e mais simples dos tipos é o *Baseline*, que requer menor poder computacional, tendo maior utilidade para dispositivos móveis. A sua taxa de compressão é alta, porém há grande perda de qualidade também.

O segundo tipo, conhecido como *Main*, é melhor que o primeiro ao aprimorar o uso da redundância temporal. É comum que seja usado para *broadcast* de TV de definição padrão.

O terceiro, conhecido como *Extended*, tem maior complexidade que os anteriores e exige mais do decodificador, com a vantagem de oferecer alta taxa de compressão, com qualidade razoável. Além disso, inclui quadros adicionais na codificação com o objetivo de simplificar a correção de erros durante a transmissão. Por essas razões, é mais utilizado para transmissão de TV pela Internet.

Por fim, o tipo *High* é utilizado para TV de alta definição. Dentre as características que o tornam o mais eficiente, está a qualidade da imagem, alcançada pela utilização de um algoritmo adaptativo [42], que escolhe entre blocos 4x4 px^2 (para áreas com muito detalhe) ou 8x8 px^2 (áreas com pouco detalhe). Essa característica, além de preservar melhor a qualidade, ainda aumenta a taxa de compressão, reduzindo a taxa de *bits*.

3.1.2 Codificação intra-quadro e inter-quadros

A codificação intra-quadro é, de forma simples, a compressão de imagens em sequência, em que não há influência de um quadro sobre a codificação de outro quadro, fazendo proveito apenas da redundância espacial.

A codificação inter-quadros, por outro lado, utiliza informações entre quadros para fazer predições, fazendo proveito da redundância temporal. A estimativa de movimento (ME) é considerada a parte essencial de quase todos os padrões de codificação de vídeo.

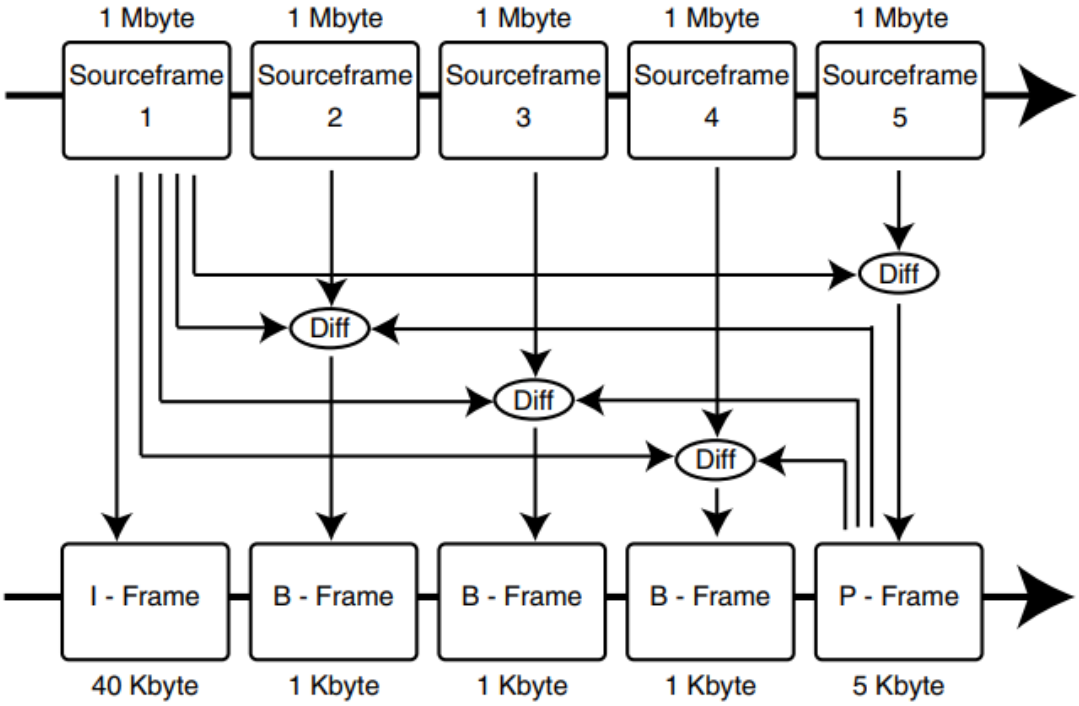
Como alguns dos *codecs* anteriores, os *codecs* mais atuais usam três tipos de quadros para compressão. Os quadros-i recebem seu nome por serem quadros intra-codificados, os quadros-b por serem quadros preditivos bidirecionais, e os quadros-p por serem quadros preditivos. Os quadros-i são também chamados quadros-chave; os quadros-b e quadros-p são também chamados quadros-delta.

Os quadros-i são submetidos apenas à codificação intra-quadro. Os quadros-p podem utilizar informação redundante de qualquer quadro-i ou quadro-p anterior. Os quadros-B podem utilizar quadros-i, quadros-b e quadros-p, passados ou futuros [43].

Como representando na *Figura 6*, a ideia básica é que um quadro-chave guarda uma imagem inteira, enquanto os quadros-delta guardam apenas as mudanças relativas a essa imagem. Na mesma figura, os *Sourceframes* se referem aos quadros originais; e *I-Frame*, *B-Frame* e *P-Frame* são respectivamente os quadros-i, quadros-b e quadros-p. Quanto aos elementos destacados *Diff*, eles consistem nas mudanças relativas entre quadros que formarão os quadros-delta. Quadros-chave podem ser inseridos no fluxo de bits sempre que, por exemplo, haja uma mudança de cena.

A partir dos quadros-chave e dos quadros-delta, são formados os grupos de imagens (GOP). Um GOP começa com um quadro-chave e termina antes do próximo quadro-chave (a próxima imagem completa). Quanto maior for o número de quadros em um GOP, maior poderá ser a qualidade do vídeo, e maior será o processamento utilizado, tanto na codificação quanto na decodificação [44].

Figura 6 – Codificação de quadros-delta



Fonte: Wootton, Cliff. A practical guide to video and audio compression, 2005, p. 193.

No H.264, fatias são segmentos de uma imagem maiores que macroblocos e menores que um quadro. Há fatias dos tipos I, P, e B. Uma fatia do tipo I é uma porção de uma imagem composta por macroblocos da mesma imagem (quadro). As fatias dos tipos P e B são seções da imagem compostas de macroblocos que podem ser de diferentes imagens (quadros) [45].

3.1.3 Funcionamento básico do H.264

As etapas de codificação do H.264 são: previsão, transformação, quantização, codificação, e encapsulamento.

A primeira etapa de codificação do H.264 é a de previsão. Um quadro do vídeo é dividido em fatias, e para cada uma dessas fatias são feitos cálculos para previsões de movimento. Isso se dá de forma temporal, usando informações de quadros já codificados, e de forma espacial, usando informações do mesmo quadro. As fatias utilizadas vão de 4x4 a 16x16. A previsão é subtraída da fatia atual para formar uma diferença, também chamada resíduo [46].

Em seguida está a etapa de transformação. A ideia é que o codificador aplica uma forma aproximada da DCT em blocos de amostras residuais. Como já comentado, essa transformada gera coeficientes que representam a distribuição de energia em cada frequência da imagem.

A próxima etapa é a de quantização, que visa manter as principais características da imagem, atenuando coeficientes obtidos na transformada pela divisão por um valor inteiro, chamado parâmetro de quantização (QP).

A consequência dessa atenuação é que alguns desses coeficientes se tornam desprezíveis. Um valor alto de QP resulta na eliminação de mais coeficientes, em maior taxa de compressão e menor qualidade da imagem, e vice-versa. É nessa etapa que a codificação causa maior perda de qualidade na imagem [47].

Posteriormente, ocorre a fase final, que é a formação do fluxo de *bits*, aproveitando dados gerados nas etapas anteriores que devem ser codificados: os coeficientes da transformada quantizados, informações para permitir que o decodificador recrie a previsão, informações sobre a estrutura dos dados comprimidos, as ferramentas de compressão usadas durante a codificação. Tendo então sido terminada a codificação, ocorre o encapsulamento (conteinerização) do fluxo. Em seguida, o fluxo é transportado aos assinantes.

No dispositivo que recebe o fluxo, começa a decodificação, basicamente, o inverso da codificação. Inicialmente, os elementos da sequência codificada são extraídos. Em seguida, as etapas de quantização e transformada são revertidos, pela multiplicação de cada coeficiente por um valor inteiro para restaurar sua escala original, e por uma transformação inversa que recria os blocos residuais.

Na fase final, o decodificador combina os blocos, adicionando à previsão os resíduos decodificados, para então reconstruir os macroblocos e, por fim, os quadros completos, que constituem o vídeo pronto para ser exibido.

3.1.4 Funcionamento básico do H.265

O princípio de funcionamento do H.265 é bem similar ao do H.264. Em comparação com seu antecessor, o H.265 também utiliza três tipos de quadros dentro de um grupo de imagens (GOP). A predição de movimento entre os quadros ocorre

com precisão muito maior e menores erros residuais [48]. Enquanto o H.264 usa 9 direções de predição, o H.265 pode usar mais de 35.

Embora cada quadro também seja particionado em fatias, no H.264 os macroblocos têm tamanho máximo de 16x16 píxeis, já no H.265 são maiores para que se possam processar em paralelo informações em unidades de árvore de codificação (CTUs) de até 64x64 píxeis [49].

As fatias são posteriormente particionadas em CTUs, que são a unidade básica de codificação e podem ter dimensões de até 64x64 píxeis. Uma CTU pode ser particionada em regiões quadradas chamadas Unidades de Codificação (CU). Cada CU é particionada em uma ou mais Unidades de Previsão (PU), e cada PU passa por um processo de previsão intra-quadro ou inter-quadros.

3.1.5 Funcionamento básico do AV1

No AV1, ocorre o particionamento de bloco, a partir de uma aprimorada estrutura em árvore com quatro partes. As imagens são particionadas em super-blocos (equivalentes a CTUs) com um tamanho máximo de 128×128. Os super-blocos podem ser sucessivamente particionados em blocos retangulares com tamanho mínimo de 4×4.

A predição interna e a predição externa no AV1 são bem aprimoradas. A predição interna pode ser de três tipos: preditor genérico direcional, preditor *Paeth*, e preditor suave. A predição externa tem acesso a até sete imagens de referência, sendo uma ou duas escolhidas por bloco. As predições de movimento acontecem em cada bloco de 8×8, e a previsão nos limites do bloco é aperfeiçoada com a utilização de preditores vizinhos [40].

O AV1 suporta três tipos de transformadas além da DCT padrão: a DST assimétrica, a DST assimétrica invertida e a transformação identidade; sobre cada uma dessas, pode ainda ser feita a transformação horizontal e a transformação vertical, isto é, são possíveis 16 transformadas. Além disso, o AV1 é bastante versátil na quantização [50].

Na codificação de entropia do AV1, que é adaptativa, vários símbolos binários são combinados em símbolos não binários, o que reduz o número de símbolos que precisam ser analisados no decodificador de entropia [40, 50].

4 COMPARATIVO DOS CODECS

Na análise de vantagens e desvantagens dos *codecs*, alguns dos critérios de comparação podem ser:

- qualidade do vídeo por taxa de *bits*;
- resoluções de imagem (por exemplo, *Full HD* e *4K*);
- velocidade e uso de *hardware* na codificação e na decodificação (por exemplo, taxa de quadros codificados por segundo);
- custos com licenciamento (como patentes);

Considerações de teor mais objetivo, como a qualidade do vídeo por taxa de *bits* e as resoluções, serão realizadas mais adiante.

Com relação aos custos com licenciamento, razões econômicas sempre são um fator de preocupação para as empresas. Por exemplo, uma empresa pode lucrar mais por prover uma melhor experiência aos utilizadores, e pode poupar custos por reduzir a taxa de *bits* da transmissão.

O H.264 é gratuito para vídeos transmitidos gratuitamente, mas qualquer uso comercial requer pagamento de taxas. No caso do H.265, como já comentado, os distribuidores de conteúdo muitas vezes não conseguem ter noção de quanto terão que pagar para utilizar o *codec*, o que dificulta sua adoção. Já no caso do padrão AV1, não há *royalties* envolvidos nem problemas com licenciamentos.

4.1 Metodologia

Para a realização dos comparativos, foram instaladas (e em alguns casos compiladas) diversas ferramentas. A primeira é o *FFmpeg*, um programa de computador utilizado para gravar, converter e fazer transmissão de áudio e vídeo.

O *FFmpeg* recebe um caminho de arquivo de entrada, um nome de arquivo de saída, e parâmetros com opções, por exemplo, o codec a ser utilizado e filtros a serem aplicados. Também permite a escolha da quantidade de processadores a

serem utilizadas, o que foi utilizado para acelerar o processo de codificação e fazer uso de um hardware robusto.

Foi também instalada uma biblioteca chamada *libvmaf* (também conhecida como filtro VMAF), auxiliar do *FFmpeg*, para a geração de relatórios dos parâmetros de qualidade SSIM, VMAF e PSNR das amostras.

Também foi necessário instalar uma ferramenta complementar ao *FFmpeg*, para que o poder computacional em paralelo fosse melhor aproveitado no caso da codificação com o *codec* ainda experimental AV1. Essa ferramenta é o *Av1an*.

Os *codecs* utilizados para codificar as amostras em H.264, H.265 e AV1 foram respectivamente a *libx264*, a *libx265* e a *libaom*. Para que pudessem ser utilizados, foi necessário baixar seu código-fonte de repositórios de livre utilização, e em seguida compilá-los.

Foi instalada também a ferramenta chamada *MediaInfo* para a obtenção de informações sobre as amostras na sua forma original. Neste caso, como apresenta uma interface gráfica, foi instalada em computador pessoal com sistema operacional *Windows*, em vez de em servidor.

As ferramentas *Microsoft Excel*, o *Google Sheets* e o *Matlab* foram usadas para o tratamento de dados (filtragem e organização), composição das tabelas e geração dos gráficos. Em vista da grande quantidade de informações geradas por essas ferramentas, foi proveitosa a criação de algumas macros para fazer operações sobre os dados.

Para que fossem processadas as amostras de vídeos, em especial com o codec AV1, um grande poder computacional foi necessário. Para atender a essa demanda, foi contratada uma instância em nuvem, de propriedade da *Google Cloud Compute* com as seguintes configurações:

- Processadores: *Intel Xeon* 2.2GHz 24-*Core* (48 *threads*) L2 *cache*: 55.0 MiB;
- Memória RAM: 96Gb;
- Sistema Operacional: *Ubuntu* 20.04.1 LTS (*Focal Fossa*) – *Kernel* 5.4.0-1021-gcp.

4.2 Descrição das amostras

Naturalmente, o conteúdo de vídeos pode ser muito diverso, por exemplo, na quantidade de movimentos dos objetos e na densidade dos seus detalhes. Diante dos diferentes conteúdos televisivos, diferentes tipos de redundância podem ser explorados e reduzidos, e os métodos utilizados e aproveitamentos diferem em cada codec.

Levando os motivos acima em consideração, para a análise comparativa da qualidade dos *codecs*, foram selecionadas 4 amostras de vídeos de uma forma que houvesse uma razoável diversidade, tornando mais justa a comparação dos *codecs*.

As amostras de vídeos brutos utilizadas são ilustradas na *Figura 7*, tendo sido obtidas de *UltraVideo* (créditos na *Figura 7*), no formato YUV e contêiner Y4M. Todos os arquivos brutos utilizados possuem a mesma duração e taxa de quadros, e assim, como nenhuma compressão foi previamente realizada sobre estes, o mesmo tamanho. A taxa de quadros dos vídeos é 30 quatros por segundo.

No primeiro vídeo, há uma pessoa realizando movimentos suaves, com os fios de cabelos balançando com o vento. No segundo vídeo, há um barco em movimento. No terceiro vídeo, há uma abelha voando por entre flores. No quarto vídeo, há uma pessoa montada a cavalo em movimento.

Figura 7 - Miniaturas das amostras utilizadas

Fonte: Montagem dos autores. Amostras originais de *UltraVideo*. Disponível em http://ultravideo.cs.tut.fi/#testsequences . Último acesso em junho de 2020.

4.3 Obtenção das métricas

Com a ferramenta *MediaInfo*, foram obtidas informações sobre as amostras: tamanho e taxa de *bits* necessária para a transmissão. A *Tabela 1* a seguir mostra um resumo das propriedades dos vídeos utilizados para os testes.

Tabela 1 – Amostras de vídeos utilizadas

Amostra	Full HD (1920 x 1080)		4K (3840x2160)	
	Tamanho (GiB)	Taxa de bits (Gbps)	Tamanho (GiB)	Taxa de bits (Gbps)
Amostra 1	1,74	0,73	6,95	2,92
Amostra 2	1,74	0,73	6,95	2,92
Amostra 3	1,74	0,73	6,95	2,92
Amostra 4	1,74	0,73	6,95	2,92

Três parâmetros objetivos de qualidade de vídeo foram determinados: PSNR, SSIM e VMAF. O tamanho e a taxa de *bits* dos vídeos posteriores à codificação

também foram determinados. Essas medidas foram realizadas para dois *presets* dos *codecs* H.264 e H.265, e para o *preset* padrão do *codec* AV1.

Com o auxílio da ferramenta *FFmpeg*, as amostras foram codificadas usando cada um dos *codecs* em duas configurações (conhecidas como *presets*). No *Anexo A*, estão algumas das linhas de comando utilizadas para codificar os vídeos.

A configuração *medium* é a padrão tanto do H.264 como do H.265. A configuração *veryfast* faz o processamento de forma mais rápida que a configuração padrão, o que faz sentido do ponto de vista de transmissões ao vivo. No caso do AV1, a escolha de configuração não se aplica.

Durante a codificação, as amostras foram analisadas utilizando a *libvmaf*, com o objetivo de determinar os parâmetros de qualidade. Esse módulo recebe duas entradas para análise: a amostra pré-codificação, que serve de referência, e a amostra pós-codificação. O módulo fornece como saída os valores de PSNR, SSIM e VMAF para cada quadro.

O tamanho e a taxa de bits dos vídeos codificados para as resoluções *Full HD* e 4K estão registrados na *Tabela 2* a seguir:

Tabela 2: Tamanho e taxa de bits após a codificação

Referência	Codec	Preset	Full HD (1920x1080)		4K (3840x2160)	
			Tamanho (MiB)	Taxa de bits (Mbps)	Tamanho (MiB)	Taxa de bits (Mbps)
Amostra 1	H.264	medium	12,70	5,21	188,00	79,10
	H.264	veryfast	10,40	4,28	196,00	82,20
	H.265	medium	2,68	1,10	13,30	5,46
	H.265	veryfast	2,59	1,06	11,90	4,89
	AV1	N/A	5,32	2,18	115,00	48,10
Amostra 2	H.264	medium	8,42	3,45	30,80	12,90
	H.264	veryfast	6,28	2,57	20,90	8,54
	H.265	medium	2,98	1,22	7,57	3,10
	H.265	veryfast	2,65	1,11	7,25	2,97
	AV1	N/A	6,26	2,57	15,30	6,28
Amostra 3	H.264	medium	2,90	1,19	24,90	10,50
	H.264	veryfast	1,30	0,53	13,80	5,67
	H.265	medium	0,53	0,22	1,50	0,62

	H.265	veryfast	0,50	0,21	1,43	0,58
	AV1	N/A	0,99	0,40	5,00	2,05
Amostra 4	H.264	medium	8,82	3,61	40,90	17,20
	H.264	veryfast	7,51	3,08	39,60	16,60
	H.265	medium	3,09	1,27	8,88	3,64
	H.265	veryfast	3,17	1,30	8,61	3,53
	AV1	N/A	5,04	2,06	16,00	6,57

Com os parâmetros PSNR, SSIM e VMAF obtidos para os vídeos codificados, foi realizado um tratamento de dados, e então foram calculados a média e o desvio padrão respectivos.

Para a resolução *Full HD* as médias dos parâmetros (e o desvio padrão) estão registrados na *Tabela 3* a seguir:

Tabela 3 - Parâmetros de análise do Full HD (1920x1080)

Referência	Codec	Preset	PSNR	SSIM	VMAF
Amostra 1 (Full HD)	H.264	medium	36,7706 ± 3,6029	0,9689 ± 0,0269	72,1388 ± 18,0301
	H.264	veryfast	36,6631 ± 3,4914	0,9683 ± 0,0256	69,0762 ± 16,7252
	H.265	medium	36,3555 ± 3,3099	0,9653 ± 0,0247	66,3697 ± 15,5245
	H.265	veryfast	36,3686 ± 3,2882	0,9655 ± 0,0243	65,3979 ± 15,2183
	AV1	N/A	39,6334 ± 0,1169	0,9883 ± 0,0007	88,0354 ± 1,5719
Amostra 2 (Full HD)	H.264	medium	39,7961 ± 3,4882	0,9819 ± 0,0192	79,8450 ± 14,3225
	H.264	veryfast	39,1877 ± 2,9795	0,9809 ± 0,0173	76,4291 ± 12,6557
	H.265	medium	38,9034 ± 2,7175	0,9789 ± 0,0166	76,1496 ± 12,4564
	H.265	veryfast	38,5507 ± 2,4945	0,9780 ± 0,0162	74,3593 ± 11,8608
	AV1	N/A	43,9825 ± 0,3736	0,9959 ± 0,0005	93,6371 ± 0,8561
Amostra 3 (Full HD)	H.264	medium	41,9480 ± 0,4644	0,9963 ± 0,0008	91,0374 ± 1,0392
	H.264	veryfast	41,1109 ± 0,4124	0,9950 ± 0,0008	87,7371 ± 1,0490
	H.265	medium	41,2869 ± 0,3810	0,9946 ± 0,0008	86,7719 ± 0,9725
	H.265	veryfast	41,1491 ± 0,3687	0,9944 ± 0,0008	86,0174 ± 0,9734
	AV1	N/A	43,2213 ± 0,4316	0,9976 ± 0,0029	94,0819 ± 2,8190
Amostra 4 (Full HD)	H.264	medium	35,4285 ± 9,4648	0,9027 ± 0,1324	65,5782 ± 37,3908
	H.264	veryfast	35,1381 ± 9,2578	0,9028 ± 0,1303	63,5952 ± 36,2569
	H.265	medium	34,8294 ± 9,0328	0,9001 ± 0,1287	62,1405 ± 35,4253
	H.265	veryfast	34,6138 ± 8,8782	0,9000 ± 0,1274	60,7044 ± 34,4248

	AV1	N/A	43,1998 ± 0,2987	0,9941 ± 0,0016	96,7890 ± 4,8529

As médias (e desvio padrão) dos parâmetros PSNR, SSIM e VMAF dos vídeos codificados para a resolução 4K estão registrados na *Tabela 4* a seguir:

Tabela 4 - Parâmetros de análise do 4K (3840x2160)

Referência	Codec	Preset	PSNR	SSIM	VMAF
Amostra 1 (4K)	H.264	medium	34,0457 ± 2,8536	0,9733 ± 0,0324	62,5623 ± 19,8889
	H.264	veryfast	34,2013 ± 2,9229	0,9732 ± 0,0318	60,7667 ± 19,0630
	H.265	medium	33,3601 ± 2,1129	0,9687 ± 0,0268	51,4919 ± 14,8534
	H.265	veryfast	33,4013 ± 2,1093	0,9693 ± 0,0264	50,5515 ± 14,4775
	AV1	N/A	36,0060 ± 0,7021	0,9949 ± 0,0014	75,9465 ± 4,5366
Amostra 2 (4K)	H.264	medium	39,7275 ± 3,6959	0,9842 ± 0,0202	72,5600 ± 20,1600
	H.264	veryfast	39,4121 ± 3,3430	0,9839 ± 0,0185	68,4178 ± 17,8169
	H.265	medium	39,0017 ± 3,1313	0,9823 ± 0,0188	67,6046 ± 17,3595
	H.265	veryfast	39,0097 ± 3,0878	0,9824 ± 0,0186	66,9832 ± 16,9876
	AV1	N/A	43,0232 ± 0,1917	0,9975 ± 0,0003	89,2791 ± 1,0231
Amostra 3 (4K)	H.264	medium	38,7543 ± 0,3234	0,9974 ± 0,0010	83,7036 ± 2,0163
	H.264	veryfast	38,5051 ± 0,2838	0,9968 ± 0,0009	81,2203 ± 1,7177
	H.265	medium	38,6205 ± 0,2322	0,9964 ± 0,0008	79,2071 ± 1,4266
	H.265	veryfast	38,6158 ± 0,2305	0,9964 ± 0,0008	78,8210 ± 1,3759
	AV1	N/A	39,1900 ± 0,0519	0,9981 ± 0,0002	85,8325 ± 0,4501
Amostra 4 (4K)	H.264	medium	33,6525 ± 8,2583	0,9047 ± 0,1347	60,7115 ± 35,8089
	H.264	veryfast	33,6123 ± 8,2201	0,9049 ± 0,1333	59,1687 ± 34,6667
	H.265	medium	33,3551 ± 8,0401	0,9029 ± 0,1321	56,5703 ± 33,3514
	H.265	veryfast	33,3494 ± 8,0323	0,9032 ± 0,1317	55,8447 ± 32,9462
	AV1	N/A	39,6966 ± 0,4377	0,9963 ± 0,0011	90,0480 ± 10,6351

A análise dos resultados obtidos com o PSNR, SSIM e VMAF demonstram que, em ambas as resoluções, o AV1 possui melhor desempenho em comparação com o H.264 e H.265, visto que ele faz o uso de técnicas mais aprimoradas em relação aos demais. Com isso, a qualidade final do vídeo é superior, por exemplo, para uma mesma taxa de *bits*.

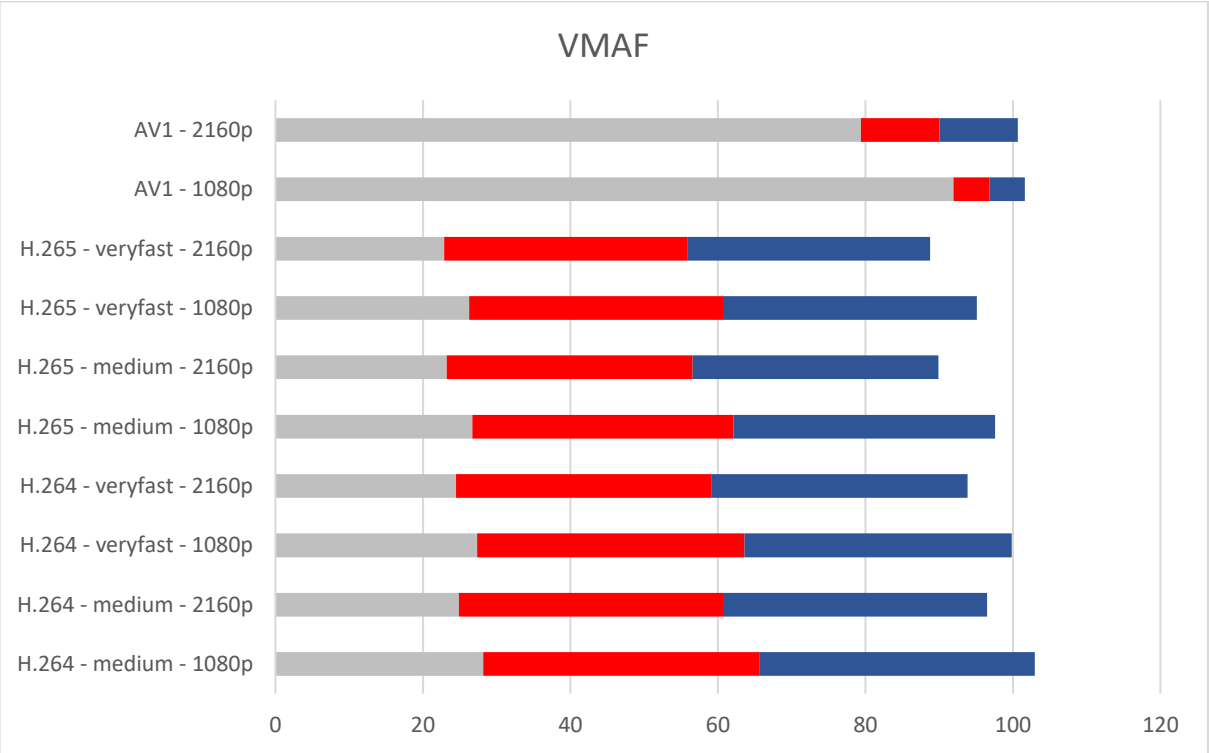
Nota-se ainda que, embora os 3 parâmetros possam apontar para conclusões semelhantes, os valores obtidos para VMAF apresentam maior

sensibilidade (estando de acordo com a bibliografia), e descrevem melhor a qualidade dos vídeos codificados.

Além disso, nota-se que há perda de qualidade para a configuração *veryfast* em comparação com a configuração *medium*, em troca de menor taxa de *bits*, o que é importante para a boa experiência do telespectador. A exceção notável é no caso do H.264 para a resolução 4K, indicando que esse *codec* não é otimizado para altas resoluções.

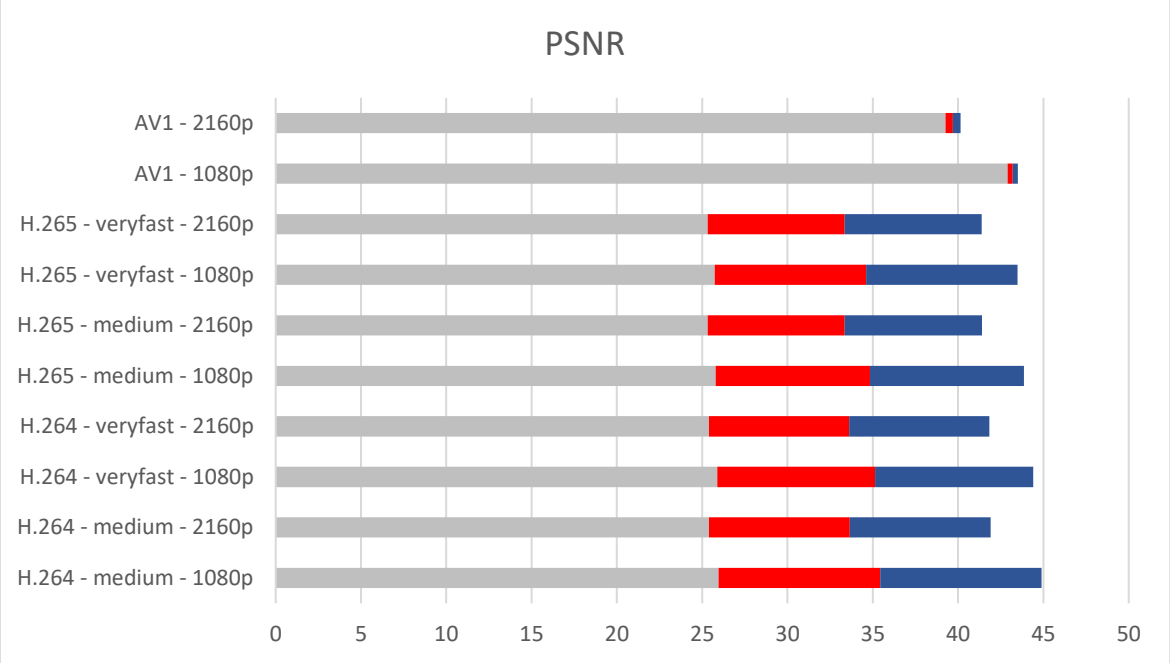
Na *Figura 8*, um gráfico mostra o parâmetro VMAF obtido para diferentes *presets* dos *codecs*, nas resoluções *Full HD* e 4K. Na *Figura 9*, o gráfico é para o parâmetro PSNR, e na *Figura 10* o parâmetro SSIM. Os gráficos foram gerados a partir dos resultados obtidos para a quarta amostra.

Figura 8 - VMAF obtido para diferentes *presets*. A área vermelha indica a variação negativa e área azul a variação positiva, estando o valor médio no seu encontro.



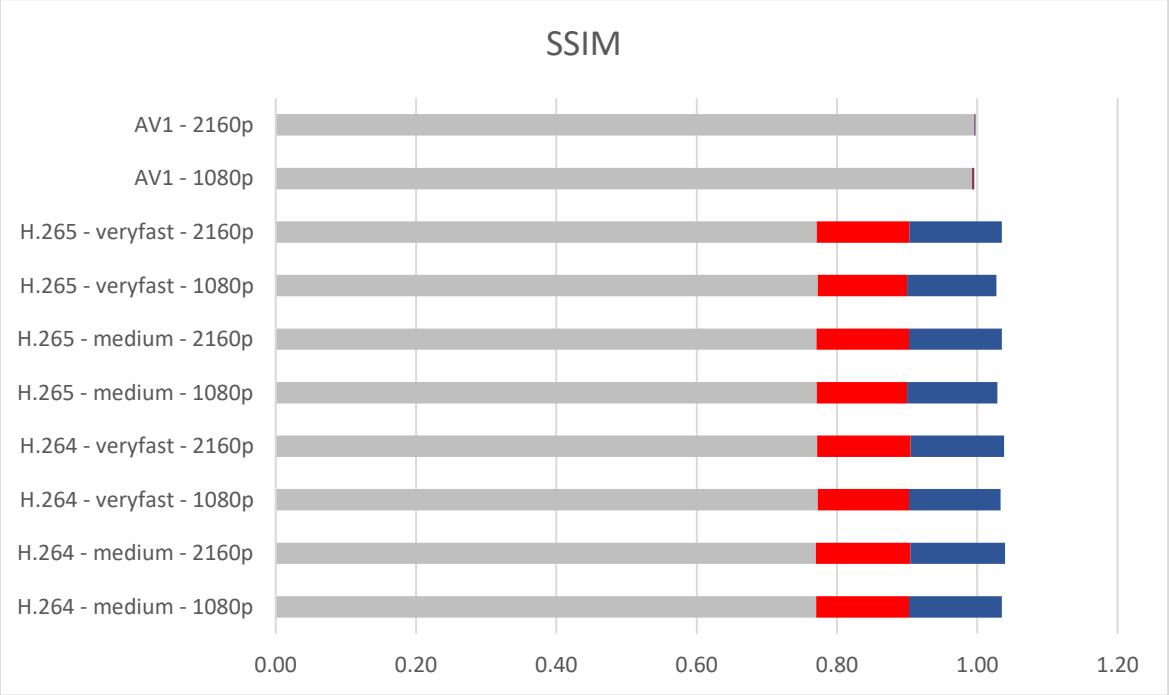
Fonte: Autoria própria.

Figura 9 - PSNR obtido para diferentes *presets*. A área vermelha indica a variação negativa e área azul a variação positiva, estando o valor médio no seu encontro.



Fonte: Autoria própria.

Figura 10 - SSIM obtido para diferentes *presets*. A área vermelha indica a variação negativa e área azul a variação positiva, estando o valor médio no seu encontro.



Fonte: Autoria própria.

Como se pode notar, o AV1 recebe destaque nas métricas de qualidade. O parâmetro VMAF é o mais preciso, conforme já mencionado, mas as outras métricas estão significativamente em concordância com os resultados obtidos de VMAF.

Como a execução das métricas é realizada quadro a quadro, são obtidos resultados diferentes para todos os *frames* do vídeo, sendo assim, vídeos com menos movimentos apresentam uma melhor qualidade quando comparados com vídeos com muitos elementos em movimento, quando usado uma mesma taxa de *bit*. Todavia, um mesmo conteúdo televisivo pode oscilar entre cenas estáticas e cenas dinâmicas, fazendo com que haja uma variação da qualidade.

É importante que o *codec* usado possua uma boa reação a mudanças de cenas para que não ocorra perdas de qualidade no vídeo. O AV1 demonstra ser um *codec* estável nesse aspecto, visto que apresenta um menor desvio padrão em todas as amostras e parâmetros quando comparado aos demais *codecs*.

5 CONCLUSÕES

Algumas considerações devem ser feitas na decisão sobre qual *codec* é mais apropriado em um determinado cenário de utilização. Eles devem ser utilizados de acordo com as necessidades, a compatibilidade, o orçamento, as necessidades de largura de banda do equipamento e muito mais.

A escolha correta do *codec* é de grande importância no desenvolvimento de um sistema de codificação de vídeo, visto que, caso opte-se por um sistema no qual necessite de licenciamento para seu uso, acarretará custos extras ao projeto. Todavia, poderão incorrer outros custos para que o processo de codificação seja realizado, como a necessidade de um maior poder computacional ou até mesmo maior demanda por tráfego de dados.

Com o tempo, houve evoluções nos métodos de compressão de vídeo, no entanto, *codecs* que compactam vídeo em tamanhos menores, mantendo mais qualidade, tendem a exigir mais poder de processamento, tanto para codificar quanto para decodificar.

O H.264 é o *codec* com maior compatibilidade, mais econômico em termos computacionais, mas que pode oferecer a menor qualidade de experiência e a menor economia em termos de largura de banda.

O H.265 se mostra altamente eficiente, mas sem grandes perspectivas por causa dos seus problemas de licenciamento. No cenário de IPTV, sua alta eficiência e qualidade de imagem são de muito proveito. Seu consumo computacional é viável para a transmissão ao vivo.

O AV1 se mostra uma grande evolução no ponto de vista de licenças, compatibilidade, eficiência. Porém, ainda está em fase experimental e é bastante lento (em troca de alta qualidade).

Por fim, considera-se que a possibilidade de gerar os vídeos em diferentes formatos é desejável do ponto de vista de compatibilidade com mais plataformas, podendo se aplicar tanto ao *codec* quanto ao contêiner. Para que haja uma maior compatibilidade, é importante que sejam oferecidos os mais diversos métodos de codificação visando diferentes públicos.

REFERÊNCIAS

[01] IBGE. **Uso de Internet, televisão e celular no Brasil**. Disponível em: https://educa.ibge.gov.br/jovens/materias-especiais/20787-uso-de-internet-televisao-e-celular-no-brasil.html. Acesso em: 4 abr. 2020.

[02] WHITAKER, Jerry. **Interactive Television Demystified**. 1. ed. [S.l.]: McGraw-Hill, 2001.

[03] PICCIONI, Carlos; BECKER, Valdecir; MONTEZ, Carlos. Uma aplicação de Governo Eletrônico Usando Televisão Digital Interativa.

[04] AGÊNCIA EBC. **Consumo de vídeo e áudio online cresce no Brasil, aponta pesquisa**. Disponível em: https://agenciabrasil.ebc.com.br/geral/noticia/2020-05/consumo-de-video-e-audio-online-cresce-no-brasil-aponta-pesquisa. Acesso em: 4 abr. 2020.

[05] NETSHOW.ME. **Por que o mercado OTT é uma tendência mundial?**. Disponível em: https://netshow.me/blog/mercado-ott/. Acesso em: 5 abr. 2020.

[06] O'DRISCOLL, Gerard. **Next Generation IPTV Services and Technologies**. 1. ed. New Jersey: Wiley, 2007.

[07] MENIN, Eyal. **Streaming Media Handbook**. 1. ed. [S.l.]: Prentice Hall PTR, 2003.

[08] CHEN, Y.et Al.. An Overview of Core Coding Tools in the AV1 Video Codec. **2018 Picture Coding Symposium (PCS)**, San Francisco, CA, v. 1, n. 1, p. 41-45, set./2018.

[09] SAKTHIVEL, M.. **Broadcasters' Rights in the Digital Era: Copyright Concerns on Live Streaming**. 1. ed. [S.l.]: BRILL, 2020.

[10] AKAMAI. **OTT Service Providers and Live Streaming Services**. Disponível em: https://www.akamai.com/us/en/multimedia/documents/white-paper/ott-service-providers-and-live-streaming-services.pdf. Acesso em: 17 mai. 2020.

[11] ZHANG, Jinyun; WANG, Yige; RONG, Bo. QoS/QoE techniques for IPTV transmissions. **2009 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting**, Bilbao, Espanha, v. 1, n. 1, p. 1-6, mai./2009.

[12] KEYSIGHT. **How network conditions impact IPTV QoE**. Disponível em: http://literature.cdn.keysight.com/litweb/pdf/5989-6143EN.pdf. Acesso em: 9 mai. 2020.

[13] WU, H. R. *et al*. **QoE Subjective and Objective Evaluation Methodologies**. 1. ed. [S.l.]: Wiley, 2015.

[14] JONNALAGADDA, Venkata Ramana; MUSTI, Vineela. Evaluation of Video Quality of Experience using EvalVid. Karlskrona, Suécia, jul./2012.

[15] HOßFELD, T. *et al*. QoE beyond the MOS: an in-depth look at QoE via better metrics and their relation to MOS. **Quality and User Experience**, v. 1, p. 1-23, set./2016.

[16] GAURAV, Rahul; ATES, Hasan Fehmi. Efficient Quality of Multimedia Experience Using Perceptual Quality Metrics. **Proceedings of 3rd International Conference on Advanced Computing, Networking and Informatics**, v. 1, p. 487-496, jan./2016.

[17] HORÉ, Alain; ZIOU, Djemel. Image Quality Metrics: PSNR vs. SSIM. **2010 20th International Conference on Pattern Recognition**, Istanbul, Turquia, v. 1, n. 1, p. 2366-2369, ago./2010.

[18] GARCÍA, B. *et al*., v. 1, n. 8, jul./2019.

[19] NAMEE, B. M. **Digital image processing: Digital image fundamentals**. Disponível em: http://www.comp.dit.ie/bmacnamee/materials/dip/lectures/ImageProcessing2-ImageProcessingFundamentals.ppt. Acesso em: 10 mai. 2020.

[20] AUSTERBERRY, D.. **The Technology of Video and Audio Streaming**. 2. ed. [S.l.]: Focal Press, 2005.

[21] GONZALEZ, R. C; WOODS, R. E.. **Digital Image Fundamentals, in Digital Image Processing**. 2. ed. Upper Saddle River: Prentice Hall, 2001.

[22] ITU-R. Parameter values for the HDTV standards for production and international programme exchange: Rec. ITU-R-BT 709, mar./2002.

[23] HANZO, Lajos; CHERRIMAN, Peter; STREIT, Jurgen. **Video Compression and Communications**: From Basics to H.261, H.263, H.264, MPEG4 for DVB and HSDPA-Style Adaptive Turbo-Transceivers. 2. ed. [S.l.]: Wiley, 2007.

[24] ISO/IEC 13818-2. International Organization for Standardization. Coding of Moving Pictures and Associated Audio – MPEG-2 Video. mai./1995.

[25] RICHARDSON, I. E. G. **H.264 and MPEG-4 Video Compression**. Chichester, UK: Wiley, 2003.

[26] MIANO, John. **Compressed Image File Formats**: JPEG, PNG, GIF, XBM, BMP. [S.l.]: Addison-Wesley Professional, 1999.

[27] SHI, Yun-qing; SUN, Huifang. **Image and Video Compression for Multimedia Engineering**: Fundamentals, Algorithms, and Standards. 2. ed. [S.l.]: CRC Press, 2008.

[28] KIM, Byung-gyu; GOSWAMI, Kalyan. **Basic Prediction Techniques in Modern Video Coding Standards**. 1. ed. [S.l.]: Springer, 2016.

[29] ROBINSON, S. J.; SCHMIDT, J. T.. Fluorescent Penetrant Sensitivity and Removability: What the Eye Can See, a Fluorometer Can Measure, Materials Evaluation. v. 42, n. 8, p. 1029-1034, jul./1984.

[30] ISO/IEC 14496-14. **Information technology — Coding of audio-visual objects — Part 14: MP4 file format**. ed. [S.l.: s.n.], 2018.

[31] WORKSPACE DIGITAL. **Video Containers and Distinguishing Between Them**. Disponível em: https://workspacedigital.com/video-containers/. Acesso em: 3 jun. 2020.

[32] RISTIC, Dan. **Learning WebRTC**: Develop interactive real-time communication applications with WebRTC. Birmingham: Packt Publishing, 2015.

[33] O'DRISCOLL, Gerard. **Next Generation IPTV Services and Technologies**. 1. ed. New Jersey: Wiley, 2007.

[34] KENCHANNAVAR, Harish; THOMAS, S.; KULKARNI, U.p.. Efficiency of FEC coding in IP networks. **Conference: Proceedings of the ICWET '10 International Conference & Workshop on Emerging Trends in Technology**, Mumbai, Maharashtra, India, p. 358-361, fev./2010.

[35] ASADUZZAMAN; GUNASEKARA, Abu And; GOVIPALAGODAGE. Power and performance analysis of multimedia applications running on low-power devices by cache modeling. **Multimedia Tools and Applications**, v. 72, set./2014.

[36] SZE, Vivienne; BUDAGAVI, Madhukar; SULLIVAN, Gary J.. **High Efficiency Video Coding (HEVC): Algorithms and Architectures**. 1. ed. [S.l.]: Springer, 2014. p. 7.

[37] BING, Benny. **Next-Generation Video Coding and Streaming**: Wiley, 2015. p. 246-247.

[38] AUWERA, G. V. D; DAVID, P. T.; REISSLEIN, M.. Traffic characteristics of H.264/AVC variable bit rate video. **IEEE Communications Magazine**, v. 46, n. 11, p. 164-174, nov./2008. Disponível em: https://ieeexplore.ieee.org/document/4689260. Acesso em: 18 mai. 2020.

[39] UHRINA, M. *et al*. Impact of H.264/AVC and H.265/HEVC CompressionStandards on the Video Quality for 4K resolution. **DIGITAL IMAGE PROCESSING AND COMPUTER GRAPHICS**, NULL, v. 12, n. 4, jan./2014. Disponível em: http://dspace5.vsb.cz/bitstream/handle/10084/112124/1216-6746-1-PB.pdf. Acesso em: 9 abr. 2020.

[40] CHEN, Y. *et al*. An Overview of Core Coding Tools in the AV1 Video Codec. **2018 Picture Coding Symposium (PCS)**, n. 10000, p. 41-45, jun./2018. Disponível em: https://ieeexplore.ieee.org/document/8456249. Acesso em: 13 mai. 2020.

[41] HELD, Gilbert. **Understanding IPTV**. ed. New York: Auerbach Publications, 2007. p. 75-76.

[42] DESIGN & REUSE. **H.264 High Profile: Codec for Broadcast & Professional Video Application**. Disponível em: https://www.design-reuse.com/articles/20776/h-264-high-profile-codec-video.html. Acesso em: 24 mai. 2020.

[43] INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES. **Overview of MPEG-1/2/4 Coding** . Disponível em: http://www.irisa.fr/armor/lesmembres/Mohamed/Thesis/node120.html. Acesso em: 3 mai. 2020.

[44] AWS ELEMENTAL LIVE. **Encoding – Group of Pictures (GOP)**. Disponível em: https://docs.aws.amazon.com/elemental-live/latest/ug/vq-gop.html. Acesso em: 13 mai. 2020.

[45] PURDUE UNIVERSITY - COLLEGE OF ENGINEERING. **Video coding with H.264/AVC: Tools, Performance, and Complexity**. Disponível em: https://engineering.purdue.edu/~zhu0/ece634_s19/lecture/h264.pdf. Acesso em: 19 mai. 2020.

[46] VCODEX. **An Overview of H.264 Advanced Video Coding**. Disponível em: https://www.vcodex.com/an-overview-of-h264-advanced-video-coding/. Acesso em: 15 mai. 2020.

[47] JMCGOWAN. **AVI Overview: Video Compression Technologies**. Disponível em: http://jmcgowan.com/avialgo.html. Acesso em: 25 mai. 2020.

[48] STANKOWSKI, J. *et al*. Analysis of the complexity of the HEVC motion estimation. **2016 International Conference on Systems, Signals and Image Processing (IWSSIP), Bratislava**, p. 1-4, jul./2016. Disponível em: https://ieeexplore.ieee.org/document/7502731. Acesso em: 28 mai. 2020.

[49] AHN *et al*. Implementation of fast HEVC encoder based on SIMD and data-level parallelism. **EURASIP Journal on Image and Video Processing**, v. 2014, p. 16, mar./2014. Disponível em: https://www.researchgate.net/publication/269588954_Implementation_of_fast_HEVC_encoder_based_on_SIMD_and_data-level_parallelism. Acesso em: 1 jun. 2020.

[50] IBC365. **AV1: Implementation, performance and applications**. Disponível em: https://www.ibc.org/av1-implementation-performance-and-applications-/3304.article. Acesso em: 3 jun. 2020.

Anexo A - Linhas de comando utilizadas para a codificação das amostras

ffmpeg -i video1-1080p.y4m -c:v libx264 -preset medium -crf 23 -c:a copy video1-1080p-medium-h264.mkv

ffmpeg -i video1-1080p-medium-h264.mkv -i video1-1080p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video1-1080p-medium-h264.txt" -f null -

ffmpeg -i video2-1080p.y4m -c:v libx264 -preset medium -crf 23 -c:a copy video2-1080p-medium-h264.mkv

ffmpeg -i video2-1080p-medium-h264.mkv -i video2-1080p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video2-1080p-medium-h264.txt" -f null -

ffmpeg -i video3-1080p.y4m -c:v libx264 -preset medium -crf 23 -c:a copy video3-1080p-medium-h264.mkv

ffmpeg -i video3-1080p-medium-h264.mkv -i video3-1080p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video3-1080p-medium-h264.txt" -f null -

ffmpeg -i video4-1080p.y4m -c:v libx264 -preset medium -crf 23 -c:a copy video4-1080p-medium-h264.mkv

ffmpeg -i video4-1080p-medium-h264.mkv -i video4-1080p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video4-1080p-medium-h264.txt" -f null -

ffmpeg -i video1-2160p.y4m -c:v libx264 -preset medium -crf 23 -c:a copy video1-2160p-medium-h264.mkv

ffmpeg -i video1-2160p-medium-h264.mkv -i video1-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video1-2160p-medium-h264.txt" -f null -

ffmpeg -i video2-2160p.y4m -c:v libx264 -preset medium -crf 23 -c:a copy video2-2160p-medium-h264.mkv

ffmpeg -i video2-2160p-medium-h264.mkv -i video2-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video2-2160p-medium-h264.txt" -f null -

ffmpeg -i video3-2160p.y4m -c:v libx264 -preset medium -crf 23 -c:a copy video3-2160p-medium-h264.mkv

ffmpeg -i video3-2160p-medium-h264.mkv -i video3-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video3-2160p-medium-h264.txt" -f null -

ffmpeg -i video4-2160p.y4m -c:v libx264 -preset medium -crf 23 -c:a copy video4-2160p-medium-h264.mkv

ffmpeg -i video4-2160p-medium-h264.mkv -i video4-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video4-2160p-medium-h264.txt" -f null -

ffmpeg -i video1-1080p.y4m -c:v libx264 -preset veryfast -crf 23 -c:a copy video1-1080p-veryfast-h264.mkv

ffmpeg -i video1-1080p-veryfast-h264.mkv -i video1-1080p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video1-1080p-veryfast-h264.txt" -f null -

ffmpeg -i video2-1080p.y4m -c:v libx264 -preset veryfast -crf 23 -c:a copy video2-1080p-veryfast-h264.mkv

ffmpeg -i video2-1080p-veryfast-h264.mkv -i video2-1080p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video2-1080p-veryfast-h264.txt" -f null -

ffmpeg -i video3-1080p.y4m -c:v libx264 -preset veryfast -crf 23 -c:a copy video3-1080p-veryfast-h264.mkv

ffmpeg -i video3-1080p-veryfast-h264.mkv -i video3-1080p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video3-1080p-veryfast-h264.txt" -f null -

ffmpeg -i video4-1080p.y4m -c:v libx264 -preset veryfast -crf 23 -c:a copy video4-1080p-veryfast-h264.mkv

ffmpeg -i video4-1080p-veryfast-h264.mkv -i video4-1080p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video4-1080p-veryfast-h264.txt" -f null -

ffmpeg -i video1-2160p.y4m -c:v libx264 -preset veryfast -crf 23 -c:a copy video1-2160p-veryfast-h264.mkv

ffmpeg -i video1-2160p-veryfast-h264.mkv -i video1-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video1-2160p-veryfast-h264.txt" -f null -

ffmpeg -i video2-2160p.y4m -c:v libx264 -preset veryfast -crf 23 -c:a copy video2-2160p-veryfast-h264.mkv

ffmpeg -i video2-2160p-veryfast-h264.mkv -i video2-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video2-2160p-veryfast-h264.txt" -f null -

ffmpeg -i video3-2160p.y4m -c:v libx264 -preset veryfast -crf 23 -c:a copy video3-2160p-veryfast-h264.mkv

ffmpeg -i video3-2160p-veryfast-h264.mkv -i video3-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video3-2160p-veryfast-h264.txt" -f null -

ffmpeg -i video4-2160p.y4m -c:v libx264 -preset veryfast -crf 23 -c:a copy video4-2160p-veryfast-h264.mkv

ffmpeg -i video4-2160p-veryfast-h264.mkv -i video4-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video4-2160p-veryfast-h264.txt" -f null -

ffmpeg -i video1-1080p.y4m -c:v libx265 -preset medium -crf 28 -c:a copy video1-1080p-medium-h265.mkv

ffmpeg -i video1-1080p-medium-h265.mkv -i video1-1080p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video1-1080p-medium-h265.txt" -f null -

ffmpeg -i video2-1080p.y4m -c:v libx265 -preset medium -crf 28 -c:a copy video2-1080p-medium-h265.mkv

ffmpeg -i video2-1080p-medium-h265.mkv -i video2-1080p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video2-1080p-medium-h265.txt" -f null -

ffmpeg -i video3-1080p.y4m -c:v libx265 -preset medium -crf 28 -c:a copy video3-1080p-medium-h265.mkv

ffmpeg -i video3-1080p-medium-h265.mkv -i video3-1080p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video3-1080p-medium-h265.txt" -f null -

ffmpeg -i video4-1080p.y4m -c:v libx265 -preset medium -crf 28 -c:a copy video4-1080p-medium-h265.mkv

ffmpeg -i video4-1080p-medium-h265.mkv -i video4-1080p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video4-1080p-medium-h265.txt" -f null -

ffmpeg -i video1-1080p.y4m -c:v libx265 -preset veryfast -crf 28 -c:a copy video1-1080p-veryfast-h265.mkv

ffmpeg -i video1-1080p-veryfast-h265.mkv -i video1-1080p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video1-1080p-veryfast-h265.txt" -f null -

ffmpeg -i video2-1080p.y4m -c:v libx265 -preset veryfast -crf 28 -c:a copy video2-1080p-veryfast-h265.mkv

ffmpeg -i video2-1080p-veryfast-h265.mkv -i video2-1080p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video2-1080p-veryfast-h265.txt" -f null -

ffmpeg -i video3-1080p.y4m -c:v libx265 -preset veryfast -crf 28 -c:a copy video3-1080p-veryfast-h265.mkv

ffmpeg -i video3-1080p-veryfast-h265.mkv -i video3-1080p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video3-1080p-veryfast-h265.txt" -f null -

ffmpeg -i video4-1080p.y4m -c:v libx265 -preset veryfast -crf 28 -c:a copy video4-1080p-veryfast-h265.mkv

ffmpeg -i video4-1080p-veryfast-h265.mkv -i video4-1080p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video4-1080p-veryfast-h265.txt" -f null -

ffmpeg -i video1-2160p.y4m -c:v libx265 -preset medium -crf 28 -c:a copy video1-2160p-medium-h265.mkv

ffmpeg -i video1-2160p-medium-h265.mkv -i video1-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video1-2160p-medium-h265.txt" -f null -

ffmpeg -i video2-2160p.y4m -c:v libx265 -preset medium -crf 28 -c:a copy video2-2160p-medium-h265.mkv

ffmpeg -i video2-2160p-medium-h265.mkv -i video2-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video2-2160p-medium-h265.txt" -f null -

ffmpeg -i video3-2160p.y4m -c:v libx265 -preset medium -crf 28 -c:a copy video3-2160p-medium-h265.mkv

ffmpeg -i video3-2160p-medium-h265.mkv -i video3-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video3-2160p-medium-h265.txt" -f null -

ffmpeg -i video4-2160p.y4m -c:v libx265 -preset medium -crf 28 -c:a copy video4-2160p-medium-h265.mkv

ffmpeg -i video4-2160p-medium-h265.mkv -i video4-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video4-2160p-medium-h265.txt" -f null -

ffmpeg -i video1-2160p.y4m -c:v libx265 -preset veryfast -crf 28 -c:a copy video1-2160p-veryfast-h265.mkv

ffmpeg -i video1-2160p-veryfast-h265.mkv -i video1-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video1-2160p-veryfast-h265.txt" -f null -

ffmpeg -i video2-2160p.y4m -c:v libx265 -preset veryfast -crf 28 -c:a copy video2-2160p-veryfast-h265.mkv

ffmpeg -i video2-2160p-veryfast-h265.mkv -i video2-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video2-2160p-veryfast-h265.txt" -f null -

ffmpeg -i video3-2160p.y4m -c:v libx265 -preset veryfast -crf 28 -c:a copy video3-2160p-veryfast-h265.mkv

ffmpeg -i video3-2160p-veryfast-h265.mkv -i video3-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video3-2160p-veryfast-h265.txt" -f null -

ffmpeg -i video4-2160p.y4m -c:v libx265 -preset veryfast -crf 28 -c:a copy video4-2160p-veryfast-h265.mkv

ffmpeg -i video4-2160p-veryfast-h265.mkv -i video4-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video4-2160p-veryfast-h265.txt" -f null -

/usr/local/bin/av1an -i video1-2160p.y4m -enc aom -v " --cpu-used=4 --end-usage=q --cq-level=30 --threads=12 " -o video1-2160p-av1-cpu4.mkv --vmaf

ffmpeg -r 30 -i video1-2160p-av1-cpu4.mkv -r 30 -i video1-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video1-2160p-av1-cpu4.txt" -f null -

/usr/local/bin/av1an -i video2-2160p.y4m -enc aom -v " --cpu-used=4 --end-usage=q --cq-level=30 --threads=12 " -o video2-2160p-av1-cpu4.mkv --vmaf

ffmpeg -r 30 -i video2-2160p-av1-cpu4.mkv -r 30 -i video2-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video2-2160p-av1-cpu4.txt" -f null -

/usr/local/bin/av1an -i video3-2160p.y4m -enc aom -v " --cpu-used=4 --end-usage=q --cq-level=30 --threads=12 " -o video3-2160p-av1-cpu4.mkv --vmaf

ffmpeg -r 30 -i video3-2160p-av1-cpu4.mkv -r 30 -i video3-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video3-2160p-av1-cpu4.txt" -f null -

/usr/local/bin/av1an -i video4-2160p.y4m -enc aom -v " --cpu-used=4 --end-usage=q --cq-level=30 --threads=12 " -o video4-2160p-av1-cpu4.mkv --vmaf

ffmpeg -r 30 -i video4-2160p-av1-cpu4.mkv -r 30 -i video4-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video4-2160p-av1-cpu4.txt" -f null -

/usr/local/bin/av1an -i video1-2160p.y4m -enc aom -v " --cpu-used=6 --end-usage=q --cq-level=30 --threads=12 " -o video1-2160p-av1-cpu6.mkv --vmaf

ffmpeg -r 30 -i video1-2160p-av1-cpu6.mkv -r 30 -i video1-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video1-2160p-av1-cpu6.txt" -f null -

/usr/local/bin/av1an -i video2-2160p.y4m -enc aom -v " --cpu-used=6 --end-usage=q --cq-level=30 --threads=12 " -o video2-2160p-av1-cpu6.mkv --vmaf

ffmpeg -r 30 -i video2-2160p-av1-cpu6.mkv -r 30 -i video2-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video2-2160p-av1-cpu6.txt" -f null -

/usr/local/bin/av1an -i video3-2160p.y4m -enc aom -v " --cpu-used=6 --end-usage=q --cq-level=30 --threads=12 " -o video3-2160p-av1-cpu6.mkv --vmaf

ffmpeg -r 30 -i video3-2160p-av1-cpu6.mkv -r 30 -i video3-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video3-2160p-av1-cpu6.txt" -f null -

/usr/local/bin/av1an -i video4-2160p.y4m -enc aom -v " --cpu-used=6 --end-usage=q --cq-level=30 --threads=12 " -o video4-2160p-av1-cpu6.mkv --vmaf

ffmpeg -r 30 -i video4-2160p-av1-cpu6.mkv -r 30 -i video4-2160p.y4m -filter_complex "[0:v][1:v]libvmaf=psnr=1:ssim=1:log_fmt=xml:log_path=video4-2160p-av1-cpu6.txt" -f null -