Universidade Fernando Pessoa

Transcodificação em tempo real de vídeo digital H.264 para *codecs* de nova geração

Joel da Silva Correia
Faculdade de Ciências e Tecnologia
Universidade Fernando Pessoa

Dissertação apresentada à Universidade Fernando Pessoa como parte dos requisitos para obtenção do grau de Mestre em Engenharia Informática no ramo de Computação Móvel

Orientador: Prof. Doutor Nuno Magalhães Ribeiro
Fevereiro 2019

Universidade Fernando Pessoa – Porto, 2019

Joel da Silva Correia

Transcodificação em tempo real de vídeo digital H.264 para *codecs* de nova geração

Universidade Fernando Pessoa – Porto, 2019

Transcodificação em tempo real de vídeo digital H.264 para *codecs* de nova geração

Por:

Joel da Silva Correia

_____

Orientador

Professor Doutor Nuno Magalhães Ribeiro

Dissertação apresentada à Universidade Fernando Pessoa como parte de requisitos para a obtenção do grau de Mestre em Engenharia Informática – Computação Móvel

Resumo

A utilização do vídeo digital tem crescido bastante nos últimos anos em diversos dispositivos. A oferta e a procura de conteúdos audiovisuais com mais qualidade devem-se muito às melhorias significativas no poder de processamento nos diversos dispositivos e ao aumento da largura de banda na Internet. A utilização de *codecs* de vídeo torna-se inevitável para tornar a transmissão de vídeo digital mais eficiente. O *codec* de vídeo H.264 continua ainda hoje a ser utilizado em diversas plataformas relacionadas com o vídeo, sendo que em alguns casos é sacrificada a qualidade de vídeo para preservar a largura de banda utilizada. Novos *codecs* de vídeo surgiram após o H.264 para ultrapassarem problemas que este não consegue solucionar face às exigências atuais.

Este trabalho analisa detalhadamente os *codecs* de vídeo H.265, VP9 e AV1 que visam substituir o H.264 no âmbito de transmissões televisivas através da implementação de um sistema apto para processar em tempo real as *streams* produzidas pelos estúdios de televisão, com o objetivo de reduzir a largura de banda necessária para a transmissão de conteúdos audiovisuais sem sacrificar a qualidade de vídeo. Propõe-se a implementação de um sistema que transcodifica as *streams* de vídeo enviadas pelos estúdios de televisão para *codecs* de vídeo mais recentes ao invés de substituir os equipamentos necessários em cada estação televisiva. Esta implementação detalha técnicas e ferramentas de *software* utilizadas num protótipo experimental, seguindo-se uma fase de testes realizados para validar o propósito da utilização deste tipo de sistema. No final, conclui-se que a utilização deste sistema permite em alguns casos reduzir o débito binário do mesmo vídeo de forma considerável, mantendo o mesmo nível de qualidade de imagem.

Palavras-chave: H.264/AVC, H.265/HEVC, VP9, AV1, Vídeo Digital, Comparação de *codecs*, Transcodificação, Televisão.

Abstract

The use of digital video has grown considerably in recent years in various devices. The supply and demand for higher-quality audiovisual content is due to significant improvements in the processing power of the various devices and the increase in bandwidth on the Internet. The use of video *codecs* becomes inevitable to make digital video transmission more efficient. The H.264 video *codec* is still used in several platforms related to the video, and in some cases the quality of video is sacrificed to preserve the bandwidth. New video *codecs* have emerged after H.264 to overcome problems that it can't solve today.

This work analyzes in detail the video *codecs* H.265, VP9 and AV1 that aim to replace the H.264 in the scope of television transmissions through the implementation of a system able to process in real time the streams produced by the television studios, with the objective to reduce the bandwidth required for the transmission of audiovisual content without sacrificing video quality. It is proposed to implement a system that transcodes video streams sent by video studios to newer video *codecs* instead of replacing the necessary equipment on each television station. This implementation details techniques and software tools used in an experimental prototype, followed by a phase of tests performed to validate the purpose of the use of this type of system. In the end, it is concluded that the use of this system allows in some cases to reduce the bitrate of the same video while maintaining the same level of image quality.

Keywords: H.264/AVC, H.265/HEVC, VP9, AV1, Digital Video, *Codec* Comparison, Transcoding, Television.

Resumé

L'utilisation de la vidéo digital a considérablement augmenté ces dernières années dans divers appareils. L'offre et la demande de contenu audiovisuel de qualité supérieure sont dues à des améliorations significatives de la puissance de traitement des différents appareils et à l'augmentation de la bande passante sur Internet. L'utilisation de *codecs* vidéo devient inévitable pour rendre la transmission vidéo numérique plus efficace. Le *codec* vidéo H.264 est toujours utilisé sur plusieurs plates-formes liées à la vidéo et, dans certains cas, la qualité de la vidéo est sacrifiée pour préserver la bande passante. De nouveaux *codecs* vidéo sont apparus après le H.264 pour résoudre des problèmes qu'il ne peut pas résoudre aujourd'hui.

Ce travail analyse les *codecs* vidéo H.265, VP9 et AV1 qui visent à remplacer le H.264 dans le domaine des transmissions télévisées par la mise en oeuvre d'un système capable de traiter en temps réel les flux produits par les studios de télévision, avec l'objectif de réduire la bande passante nécessaire à la transmission de contenu audiovisuel sans sacrifier la qualité vidéo. Il est proposé de mettre en œuvre un système permettant de transcoder les flux vidéo envoyés par les studios vidéo vers des *codecs* vidéo plus récents au lieu de remplacer l'équipement nécessaire sur chaque station de télévision. Cette mise en œuvre détaille les techniques et les outils logiciels utilisés dans un prototype expérimental, suivis d'une phase de tests permettant de valider l'objectif de l'utilisation de ce type de système. En fin, il est conclu que l'utilisation de ce système permet dans certains cas de réduire le débit binaire de la même vidéo de manière à maintenir le même niveau de qualité d'image.

Mots-clés: H.264/AVC, H.265/HEVC, VP9, AV1, Vidéo numérique, Comparaison de *codecs*, Transcodage, Télévision.

Agradecimentos

Quero agradecer ao meu orientador, professor Nuno Ribeiro, pela disponibilidade e paciência que teve comigo durante todo o processo envolvente a este trabalho.

Também quero agradecer aos professores e colegas da Universidade que me ajudaram e motivaram direto e indiretamente na realização desta dissertação.

Por último, agradeço aos meus pais por me terem proporcionado esta oportunidade de fechar mais um ciclo na vida e finalmente, à minha namorada por me ter proporcionado todo o apoio psicológico necessário durantes estes meses.

A todos, um grande obrigado.

Índice

Índice de figuras

Índice de tabelas

Acrónimos

ADSL Asymmetric Digital Subscriber Line

AOM Alliance for Open Media

AV1 AOMedia Video 1

AVC Advanced Video Coding

CABAC Context-adaptive binary arithmetic coding

CAVLC Context-adaptive variable-length

CD Compact Disk

CRT Cathode-Ray Tube

CTB Coding Tree Block

CU coding unit

DCT Discrete Cosine Transform

DVB Digital Video Broadcasting

DVD Digital Video Disc

EBU European Broadcasting Union

GOP Group of Pictures

HD High Definition

HDR High Dynamic Range

HEVC High Efficiency Video Coding

HLS HTTP Live Streaming

HTTP Hypertext Transfer Protocol

HVS Human Visual System

IETF Internet Engineering Task Force

IPTV Internet Protocol Television

ITU International Telecommunications Union

JPEG Joint Photographic Experts Group

JVT Joint Video Team

LCD Liquid Crystal Display

MOS Mean Opinion Score

MPEG Moving Picture Experts Group

MV Motion Vector

NTSC National Television System Committee

OLED Organic Light-Emitting Diode

PAL Phase Alternating Line

PNG Portable Network Graphics

PSNR Peak Signal-to-Noise Ratio

PU Prediction Unit

QP Quantization Parameter

QoE Quality of Experience

RTCP Real-Time Transport Control Protocol

RTP Real-time Transport Protocol

RTSP Real Time Streaming Protocol

SD Standard Definition

SECAM Séquentiel Couleur à Mémoire

SSIM Structural Similarity

TCP Transmission Control Protocol

TDT Televisão Digital Terrestre

UDP User Datagram Protocol

UHD Ultra High Definition

VMAF Video Multi-Method Assessment Fusion

VOD Video on Demand

VQM Video Quality Model

1. Introdução

Nos últimos anos, o vídeo digital na Internet tem tido um grande impacto na transmissão da informação. Segundo a Cisco (Cisco, 2015), por volta de 2019, entre 80 e 90% do tráfego global na Internet será constituído por *streams* de vídeo digital. É muito fácil compreender porque é que este tipo de media requer elevadas quantidades de largura de banda: trata-se de um tipo de media não-estruturado que é constituído por sequências de *frames*, contendo milhares de pixéis, que devem ser reproduzidas em tempo real de acordo com o *frame* rate definido (por exemplo, 25 fps – *frames* por segundo). No contexto da computação móvel, incluindo dispositivos tais como smartphones, tablets e computadores portáteis, tem-se vindo a assistir a um crescimento exponencial na respetiva utilização. Esta evolução verifica-se tanto em quantidade como em qualidade. Atualmente, um dispositivo móvel de gama média tem poder computacional suficiente para executar as mais variadíssimas aplicações, incluindo a capacidade de gravação e reprodução de vídeo digital. A transmissão de programas da televisão na rede tem igualmente um papel fundamental no impacte que a dimensão do vídeo digital possui sobre a largura de banda de rede que está disponível. De facto, a transmissão dos canais televisivos tem vindo a ser feita por IPTV (*Internet Protocol Television)* e cada vez menos por satélite. Os serviços de *streaming* (tais como, entre outros, a Netflix e o Amazon Prime) estão a implantar-se cada vez mais na Internet, proporcionando maior quantidade e melhor qualidade de imagem.

1.1. Descrição do problema

Este trabalho está inserido no contexto da área de especialização em multimédia e redes móveis com o intuito de contribuir para o desenvolvimento de novas técnicas que mudem a forma como é efetuada a adaptação do vídeo digital consoante a largura de banda disponível em cada momento do lado do recetor. A ideia fundamental é a de evitar o *downscaling*[1] da imagem, isto é, preservar a resolução original das *frames* do vídeo mas modificando totalmente os algoritmos de codificação do conteúdo de vídeo

[1] *Downscaling* é o processo de conversão de um sinal de vídeo digital de uma resolução maior para uma resolução mais baixa.

digital de modo a diminuir bastante o débito binário (*bitrate*), mantendo, portanto, a mesma qualidade de imagem (ou resolução).

A este respeito, o ideal seria que, do lado das estações televisivas, o vídeo fosse já gravado e codificado diretamente com o *codec* HEVC (*High Efficiency Video Coding*) e, posteriormente, transmitido nesse formato. O problema principal encontra-se exatamente nesta realidade: as câmaras de vídeo digital existentes atualmente, e que suportam a codificação de vídeo digital em HEVC, são ainda muito escassas e, além do mais, o investimento para atualizar o equipamento existente poderia não fazer qualquer sentido. Neste momento, todas as estações de televisão responsáveis pelos canais que são disponibilizados à população Portuguesa teriam que se atualizar a nível de equipamentos para implementarem uma melhor qualidade de imagem (isto é, com maior resolução) com um custo inferior no que diz respeito à largura de banda despendida.

Vários grupos estão já a adotar novas formas de codificar o vídeo digital para tornar a transmissão da respetiva *stream* mais eficiente. Por exemplo, a Google desenvolveu o *codec* VP9 (similar ao HEVC) para transmitir os vídeos na plataforma YouTube para que os utilizadores não necessitem de uma largura de banda muito elevada para reproduzirem os conteúdos de vídeo digital em alta definição. Por outro lado, a Netflix e a Amazon estão a adotar o HEVC para transmitirem conteúdos de vídeo digital gravado na resolução 4K. Mais uma vez, o objetivo é semelhante ao da Google, isto é, transmitir de forma mais eficiente os conteúdos de vídeo digital de alta resolução.

A diferença entre estas iniciativas no que diz respeito ao trabalho aqui apresentado está no lado do conteúdo a ser reproduzido. Nessas plataformas, todo o conteúdo está armazenado em disco rígido e encontra-se gravado em vários formatos (quer em termos de resolução quer do *codec* utilizado). Por exemplo, no YouTube, ao selecionar-se uma determinada resolução X ou Y, na realidade está a selecionar-se um ficheiro X ou Y. Neste contexto, este projeto pretende trabalhar com conteúdo audiovisual digital que está a ser transmitido ao vivo, como é o caso dos canais de televisão. Dado que estão a ser gravados num determinado momento recorrendo a um determinado *codec*, é precisamente esse *codec* que deverá ser utilizado para gerar a *stream* de vídeo transmitido para as operadoras e, posteriormente, para os clientes. O objetivo deste trabalho é o de combinar as tecnologias adotadas por empresas tecnológicas tais como a

Google, a Netflix e a Amazon, com transmissões em direto, efetuando, deste modo, o *transcoding* da *stream* de vídeo que está a ser transmitido em tempo real.

Por experiência própria, na qualidade de cliente de uma operadora de telecomunicações que fornece um contrato de Televisão, Internet e Telefone fixo, existe a possibilidade de verificar que tal operadora fornece débitos binários de largura de banda de Internet que não ultrapassam 6 Mbps de *download* e 1 Mbps de *upload* devido a limitações físicas da tecnologia utilizada pela operadora. Tendo em conta que a transmissão da televisão é feita por IPTV (sinal de vídeo transmitido sobre a Internet), esses valores não são claramente suficientes para que o cliente consiga visualizar canais HD (*High Definition*), uma vez que, em média, uma *stream* HD aloca entre 4 a 10 Mbps da largura de banda. Assim, como resultado final, temos que o cliente apenas poderá usufruir de canais SD (*Standard Definition*) na sua grelha de canais televisivos.

Na realidade, o tráfego na Internet gerado pelos vídeos digitais não está otimizado da melhor forma tendo em conta os novos *codecs* de vídeo que se começaram a utilizar na atualidade. As operadoras e outras plataformas de *streaming* ainda continuam a efetuar o *downscaling* da imagem para reduzir o débito binário como se o único *codec* existente ainda fosse o H.264, o que não corresponde, de todo, à realidade do desenvolvimento tecnológico atual.

Por outro lado, é importante realçar que é um facto que o poder computacional tem aumentado exponencialmente ao longo dos anos (Waldrop, 2016), fazendo com que operações computacionais que, antes, eram consideradas complexas, tais como as operações que são implementadas nos algoritmos de compressão associados aos *codecs* de vídeo mais eficientes e sofisticados, são agora possíveis numa grande variedade de dispositivos informáticos.

1.2. Objetivos do trabalho

O presente trabalho foi realizado com três objetivos principais. O primeiro prende-se com questões económicas: a aplicação do processo de *transcoding* sugerido neste trabalho poderá evitar que as estações de televisão necessitem de investir mais recursos financeiros na atualização dos respetivos equipamentos para suportarem, de raiz, a codificação dos seus programas recorrendo aos *codecs* de nova geração, tais como o HEVC e o VP9. De facto, as operadoras responsáveis pela transmissão dos canais para

o consumidor final necessitariam apenas de realizar o processo de *transcoding*, sem ser necessário modificar o processo já existente de passagem do sinal de vídeo digital entre a estação de televisão e a operadora que o transmite para a Internet. Neste contexto, o primeiro objetivo é verificar se um processo de *transcoding* tal como o que se propõe neste trabalho é viável para minimizar os recursos financeiros da transição para o novo *codec*.

O segundo objetivo diz respeito à largura de banda disponível nas operadoras. Com efeito, caso não se efetue qualquer modificação a nível de codificação de vídeo digital nos próximos anos, a rede tenderá a ficar muito sobrecarregada apenas com o *streaming* de vídeo. Recentemente, a Google criou o seu próprio *codec*, designado por VP9, precisamente para substituir a codificação de vídeo digital em H.264 nos vídeos disponibilizados no seu serviço YouTube já com o objetivo de reduzir a largura de banda necessária para transmitir e reproduzir um conteúdo de vídeo digital em tempo real a partir do YouTube. Neste caso, o segundo objetivo deste trabalho é o de efetuar um estudo aprofundado dos conceitos de codificação de vídeo digital e da respetiva aplicação em *codecs* de nova geração com o intuito de verificar se, efetivamente, diminuem a largura de banda exigida para o *streaming* de vídeo digital.

O terceiro e último objetivo que serve de motivação para o trabalho aqui descrito é o de proporcionar ao consumidor final uma qualidade de imagem mais elevada (com maior resolução) sem ser necessário exigir maior largura de banda do que aquela que é utilizada atualmente para o *streaming* de vídeo digital. Em muitos casos, o que sucede do lado das operadoras é a aplicação de restrições à qualidade da imagem, nomeadamente à resolução, como tem vindo a ser apontado, devido à limitação da largura de banda. Neste contexto, o terceiro objetivo deste trabalho é o de verificar se é possível evitar a perda da qualidade mencionada aplicando um processo de *transcoding* para um *codec* de nova geração tal como o que aqui se propõe.

1.3. Estrutura do documento

Esta dissertação inicia-se com uma introdução a este trabalho, identificando o tema tratado e colocando-o em perspetiva, referindo a respetiva importância no quotidiano do utilizador, finalizando-se com uma explicação da motivação para a sua realização e os objetivos principais que se pretende atingir.

No Capítulo 2 efetua-se uma breve análise da evolução da utilização do vídeo digital em difusão de sinal TV e na *Web*. Faz-se ainda uma introdução que explica a difusão do sinal em ambas as plataformas e a importância dos vários *codecs* de vídeo digital nesses contextos. De seguida, o Capítulo 2 efetua revisão dos vários conceitos discutidos ao longo da dissertação, desde os mais elementares, tais como os conceitos e características do vídeo digital até aos algoritmos e *codecs* de compressão de vídeo digital, discutindo e analisando os principais *codecs* utilizados atualmente, incluindo o AVC, o HEVC e o VP9, bem como futuros *codecs*, ainda em desenvolvimento, como é o caso do AV1. O grau de detalhe e especificação na análise dos *codecs* é maior nos casos do HEVC e do AV1, visto serem os *codecs* mais inovadores e mais recentes a serem adotados.

Após a revisão bibliográfica, no Capítulo 3 faz-se uma análise do Sistema cujo desenvolvimento se propõe neste trabalho, incluindo as respetivas especificações de arquitetura bem como os requisitos. São ainda descritos os principais módulos projetados para o Sistema, indicando os componentes que os integram, bem como os seus objetivos no contexto do Sistema de *transcoding* e as respetivas contribuições para os objetivos deste trabalho. No final deste capítulo são apresentados alguns casos de uso que identificam os contextos em que o Sistema aqui proposto pode ser utilizado.

De seguida, no Capítulo 4, descreve-se detalhadamente todas as ferramentas (*software* e *hardware*) necessárias para a implementação do protótipo do Sistema concebido no capítulo 3. As especificações estão divididas nos três módulos principais que constituem o sistema proposto: Servidor de *streaming*, *Transcoder* e, finalmente, o módulo dos recetores.

O Capítulo 5 expõe os testes realizados aos *codecs* estudados neste trabalho. Inicia-se com uma apresentação de alguns métodos para avaliar a qualidade de vídeo e o *software* existente para os aplicar. Recorrendo ao protótipo desenvolvido realizou-se um conjunto de testes para avaliar a viabilidade do Sistema de *transcoding* para atingir os objetivos propostos neste trabalho. São analisados os gráficos dos resultados obtidos para cada *codec* juntamente com uma breve análise do resultado que foi alcançado. É ainda abordada a situação atual do *codec* AV1 e os resultados que alcança, mesmo estando numa fase de desenvolvimento inicial.

Esta dissertação encerra-se com o Capítulo 6 onde se efetua uma análise crítica do trabalho desenvolvido no âmbito deste projeto, bem como dos resultados obtidos. No final do capítulo são sugeridas algumas modificações ao Sistema proposto no contexto do trabalho futuro, incluindo uma análise de possíveis melhorias do Sistema e a forma como se poderá acrescentar suporte para o novo *codec* AV1.

2. Estado da Arte

Neste capítulo revê-se um conjunto de conceitos importantes para a realização e compreensão do trabalho descrito nesta dissertação. O capítulo divide-se em três partes distintas: a primeira é de cariz mais introdutório, revendo a importância do vídeo digital, um pouco da sua história e analisando as respetivas características principais. Na segunda parte deste capítulo aborda-se a evolução da utilização do vídeo digital no contexto da difusão terrestre do sinal digital de TV e também na Web. Esta parte encontra-se subdividida em duas fases distintas: numa primeira fase efetua-se a abordagem do processo de difusão de canais TV por diversas plataformas e, na seguinte, apresenta-se uma explicação do processo de *streaming* de vídeo na Web, da qualidade da rede a nível nacional e internacional e de algumas previsões para o futuro. Em ambas as fases efetua-se uma análise da evolução da utilização dos *codecs* abordados no âmbito deste trabalho. A terceira e última parte, talvez a mais detalhada e extensa deste capítulo, aborda a compressão do vídeo digital, descrevendo os objetivos principais dos *codecs* de vídeo, alguns dos conceitos utilizados nesse processo e, por fim, a apresentação dos *codecs* mais importantes da atualidade, mencionando as respetivas vantagens e desvantagens, as respetivas especificações técnicas, detalhando comparações relevantes e apontando trabalhos futuros sobre as tecnologias revistas.

2.1. Evolução da utilização de vídeo digital na *Web*

A Internet serviu de base para a divulgação dos diferentes tipos de *media* que constituem os documentos e aplicações multimédia, tais como texto, áudio, imagem e vídeo. Isto levou a que houvesse a necessidade de se proceder a uma reestruturação profunda na forma como se encarava a partilha da informação pelas pessoas. Por exemplo, as músicas eram normalmente vendidas sob a forma de armazenamento físico em Compact Discs (CD-DA) e o mesmo modelo era aplicado aos filmes que eram vendidos em *tapes* VHS e, posteriormente, em suporte ótico DVD e, posteriormente, Blu-ray. Com o desenvolvimento da Internet, novos modelos tiveram que ser concebidos e adotados para que a transição para a distribuição de conteúdos digitais *online* corresse da melhor forma. O vídeo digital é o tipo de *media* que mais impacto teve no aumento da utilização da Internet em diversas formas, incluindo, entre outras, as

comunicações recorrendo a vídeo e o *Video On Demand* (VOD)[2]. Existem duas formas fundamentais de enviar um vídeo digital: por via da transferência de ficheiros (*File Download*) ou por via de um fluxo de *media*, mais conhecido por s*treaming*.

Transferir um ficheiro na forma de vídeo para o dispositivo local e depois reproduzi-lo é a técnica mais básica de transmissão deste tipo de *media*. Em termos práticos apenas é necessário um servidor *Web* de ficheiros para armazenar os vídeos e através de protocolos de rede, tais como HTTP ou FTP, realiza-se a transferência do conteúdo. Apesar da facilidade da implementação, esta aproximação possuí, no entanto, várias desvantagens. O vídeo é considerado, na *Web*, um ficheiro de grandes dimensões a nível do armazenamento, fazendo com que seja necessária uma ligação à Internet com uma largura de banda considerável para que o tempo de transferência não seja muito elevado, além de que o utilizador deve ter em consideração que vai necessitar de espaço disponível em disco para armazenar o conteúdo que descarregou. Este método envolve igualmente certos aspetos legais, incluindo, por exemplo, um simples facto: mesmo que o utilizador tenha adquirido uma cópia legal do vídeo, não há nenhum mecanismo que impeça que o ficheiro descarregado possa ser partilhado com outras pessoas, desprezando completamente os respetivos direitos de autor.

A transmissão do vídeo através de *streaming* surgiu principalmente para superar estas dificuldades e inovar a forma de distribuição *online* deste tipo de *media*. O conceito geral de *streaming* de vídeo passa por existir um servidor dedicado à tarefa de repartir o vídeo, adquirido ao vivo ou guardado localmente, em partes ou blocos que são transmitidos sequencialmente para o recetor que, posteriormente, os descodifica e os reproduz conforme são recebidos, sem existir a necessidade de se esperar pelo conteúdo completo para ser reproduzido. A Figura 2.1 ilustra a arquitetura associada ao conceito de *streaming*.

Na realidade, o *streaming* veio revolucionar o mercado do vídeo digital distribuído na *Web*, fazendo com que o utilizador não fique com o conteúdo armazenado localmente, não obrigando a infringir qualquer lei e evitando a preocupação com o respetivo armazenamento, visto que as partes transmitidas são guardadas numa memória do tipo *buffer* e, posteriormente, são descartadas à medida que foram sendo visualizadas. A

[2] VOD (*Video On Demand)* é um sistema que permite aos utilizadores selecionar conteúdo multimédia como vídeo e aúdio, podendo manipular essas fontes em qualquer parte do dia, ao invés da típica televisão via *broadcast* com programação definida.

largura de banda necessária continua a ser a mesma que é necessária para o método de *download*, visto que, apesar da forma de transmissão ser diferente, a dimensão do vídeo continua a ser a mesma.

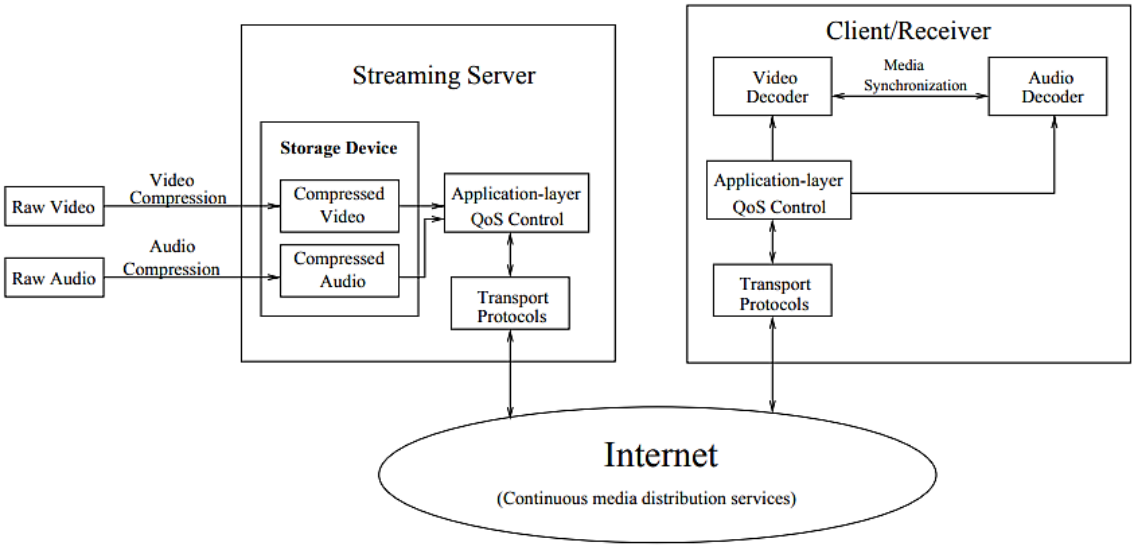

Figura 2.1 - Arquitetura de um serviço de *streaming* de vídeo sobre a internet (Wu, Member, Hou, & Zhu, 2001)

Houve, no entanto, a necessidade de criar um outro método designado por *adaptive streaming* (fluxo de *media* adaptado) que permite adaptar o ritmo de transmissão do vídeo à velocidade de ligação à Internet do recetor final. Mas esta tecnologia apenas funcionará corretamente se for regida por regras e protocolos, uma vez que o *streaming* de vídeo sobre a Internet tem que lidar com variáveis imprevisíveis e dinâmicas que incluem as seguintes:

- A largura de banda disponível num determinado instante;
- Variações no atraso da ligação;
- A taxa de perdas de *frames*.

A largura de banda e o atraso entre dois pontos na Internet é geralmente imprevisível devido ao tráfego que existe no meio. Se o emissor enviar o vídeo de forma mais rápida do que o recetor capaz de a descodificar, cria-se, automaticamente, uma congestão de dados, fazendo com que ocorram perdas de pacotes e um degradamento da qualidade do vídeo na receção. A existência desta quantidade de tráfego multimédia fez com que os investigadores sentissem a necessidade de conceber um conjunto de protocolos de

camada aplicacional dedicados à transmissão em tempo real de dados sob a forma de *streaming*, incluindo, por exemplo, o *Real-Time Transmission Protocol* (RTP), o *Real-Time Transmission Control Protocol* (RTCP) e o *Real-Time Streaming Protocol* (RTSP). O primeiro foi desenvolvido pelo grupo *Internet Engineering Task Force* (IETF) por volta de 1996 e tornou-se numa das primeiras normas na Internet com o objetivo de transportar informação em tempo real incluindo áudio e vídeo digital.

Tipicamente, o protocolo RTP está sempre associado com o seu par RTCP, o primeiro é usado para a troca de dados multimédia enquanto que o segundo é responsável por dados de controlo que proporcionam *feedback* sobre a ligação e a qualidade de serviço. Geralmente, na camada de transporte, estes pacotes são encapsulados com o protocolo UDP que apenas tem como missão entregar os pacotes sem qualquer garantia de que o pacote chegue ao destino final. Isto faz com que seja muito utilizado em aplicações tais como chamadas de voz e *streaming* de vídeo em tempo real, nas quais a latência deverá ser a menor possível sendo que este é o fator mais relevante numa ligação deste género. Pelo contrário, o TCP é igualmente um protocolo de camada de transporte, mas este possuí mecanismos para a deteção de perda de pacotes e para retransmiti-los de forma ordenada, o que faz com que possa ser muito útil em aplicações em que a latência e a variação do *jitter* não sejam fatores muito relevantes, tais como o *Video On Demand*, que requer que seja garantido que os pacotes chegam ao destino para uma experiência positiva.

O RTSP funciona sobre o TCP (mas também pode ser usado no UDP) e é utilizado para estabelecer e controlar *streams* de áudio e vídeo entre servidores multimédia e o cliente. A transmissão da *stream* de dados não é a tarefa principal do protocolo sendo que esta transmissão pode ser feita juntamente com o RTP, como se pode observar na Figura 2.2.

Como alguns autores referem (H. Schulzrinne, Rao, & Lanphier, 1998), o protocolo RTSP atua como se fosse um controlo remoto de rede para servidores multimédia, visto que possui, entre outras, a capacidade de pausar o fluxo e, reposicioná-lo para qualquer ponto no tempo.

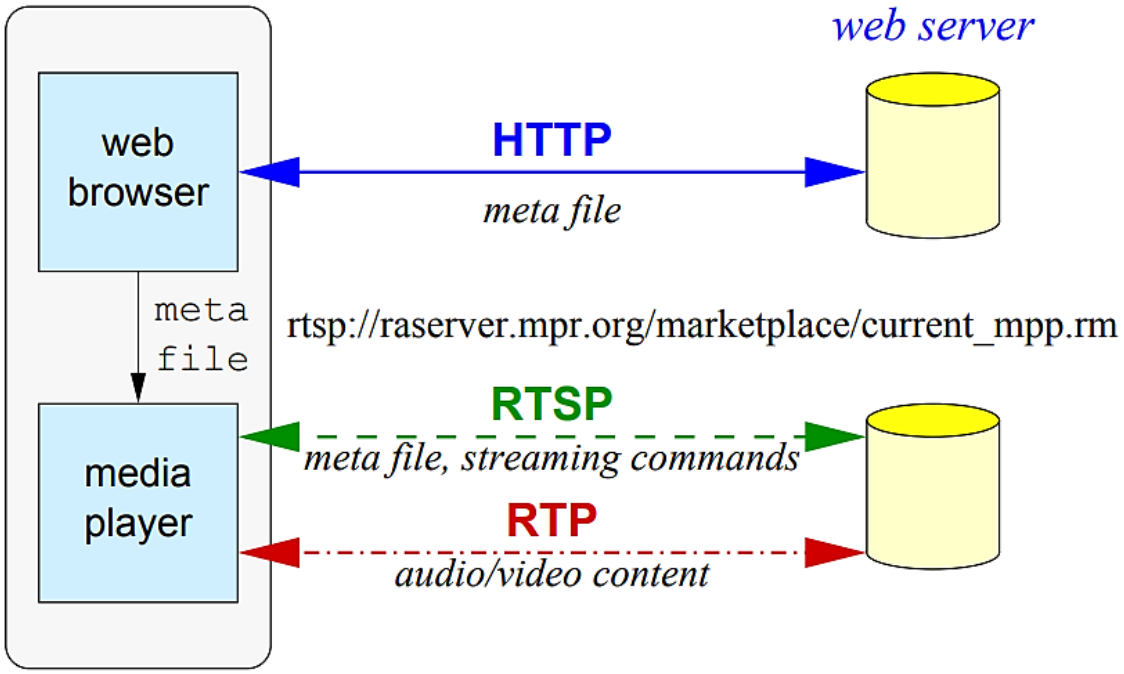

Figura 2.2 - Stream sobre o protocolo RTSP (Henning Schulzrinne, 2001)

Hoje em dia há novas técnicas que já são orientadas a todos os equipamentos desde televisões a dispositivos móveis que possuam uma ligação à Internet. O *HTTP Live Streaming* (HLS) e o *Dynamic Adaptive Streaming* over HTTP (também conhecido por MPEG-DASH) são dois exemplos que utilizam o protocolo HTTP a nível aplicacional fazendo com que não existam problemas de rejeição em muitas *firewalls* ao contrário das *streams* RTP que funcionam sobre UDP que, em muitos casos, são bloqueadas em certas portas dos *routers*. Tanto o HLS como o MPEG-DASH são tecnologias muito utilizadas na atualidade, sendo que o HLS foi, em 2017, considerado o formato de *streaming* mais popular, seguido pelo MPEG-DASH (Lederer & Lederer, 2017). Para além disso, são tecnologias que suportam de forma fiável fluxos adaptáveis (*adaptive streaming*) e também os *codecs* mais eficientes da atualidade como é o caso do H.264/265 e do VP9.

Um dos aspetos mais importantes relacionados com a utilização de vídeo digital na *Web* é a respetiva eficiência de transmissão sobre a rede. Neste contexto, abordou-se já um conjunto de técnicas e protocolos para transportar o conteúdo, mas é importante notar que, se esse conteúdo não for codificado da forma mais eficaz possível, acaba por ter que se utilizar uma largura de banda maior do que aquela que seria estritamente necessária. Para resolver este problema é fundamental recorrer-se aos *codecs* de vídeo, sendo cada vez mais necessária a sua utilização se atendermos ao aumento da qualidade

da imagem (com resoluções cada vez maiores), e ao aumento do *frame rate* que implica um aumento da taxa de fotogramas por segundo e, inclusivamente, com o aumento da profundidade de cor nos vídeos do tipo *High Dynamic Range* (HDR).

Os primeiros algoritmos de codificação de vídeo na *Web* eram fundamentalmente utilizados no contexto de tecnologias de telecomunicações, tais como a videoconferência, uma vez que as plataformas de partilhas de vídeo não eram, naquela altura, muito populares (W. Barz & A. Bassett, 2016). Em 1988 o ITU-T[3] publicou a especificação do *codec* H.261 que foi o primeiro, em termos práticos, a ser utilizado para as tarefas básicas que se podiam efetuar na Internet com o vídeo digital. Juntamente com o MPEG, estes dois grupos de trabalho elaboraram em conjunto várias especificações, cada mais vez eficientes, para *codecs* de vídeo digital. A geração que se seguiu (incluindo as normas H.262/MPEG-2 Part 2) obteve um grande sucesso no contexto das transmissões televisivas (Practice, 2011) e, nos anos que se seguiram, o formato MPEG-4 Part 2, juntamente com o *codec* H.263, começou a ser utilizado em várias aplicações de vídeo na Internet dado que proporcionava melhor desempenho na compressão do vídeo quando comparado com o seu antecessor (H.261). De facto, o H.263 obteve uma adoção relativamente rápida com a respetiva incorporação no formato FLV (*Flash Video*) que era utilizado em sítios *Web* que recorressem à ferramenta da *Adobe Systems* para reproduzir sequências de vídeo digital. No entanto, o ITU-T não era a única organização a trabalhar no domínio do desenvolvimento de algoritmos de codificação de vídeo para a *Web*, tendo na *On2 Technologies* o seu concorrente principal, tendo desenvolvido os *codecs* VP6 e VP7 que foram igualmente incorporados no *Flash Player* da *Adobe Systems*. No período temporal entre 2003 e 2006 viria a assistir-se à introdução de várias inovações tecnológicas no que dizia respeito à utilização de vídeo digital na Internet. Por exemplo, o YouTube surgiu como um serviço de partilha generalizada de vídeos na *Web*, tendo-se tornado, quando foi adquirido pela Google, na maior plataforma de partilha de vídeos do mundo (Ha, 2018). Dois anos antes do aparecimento dessa plataforma, o ITU-T tinha publicado mais uma especificação de um *codec* de vídeo: o H.264. Publicada em 2003, esta norma de codificação de vídeo digital evidenciava um aumento de eficiência da ordem dos 50% quando comparada com a norma H.263, como se analisa com mais detalhe no Capítulo

[3] O *ITU Telecommunication Standardization Sector* faz parte de um dos três setores do *International Telecommunication Union* (ITU) que tem como função coordenar as normas de telecomunicações no mundo.

2.4. Por este motivo, a norma H.264, passou a ser utilizada por um grande número de aplicações que incluem, entre muitas outras os *browsers Web* e a transmissão de televisão digital.

Contudo, apesar de passar a ser o *codec* de vídeo dominante em praticamente todas as tecnologias que envolvessem a utilização de vídeo digital, o *codec* H.264 possuía algumas desvantagens, principalmente a nível financeiro, visto que a sua utilização, em alguns casos, exigia o pagamento de *royalties*[4]. Com o aparecimento do HTML5[5] deixou de ser necessário o uso de *plugins Web* como o *Flash Player* para reproduzir vídeos nas páginas *Web*, uma vez que o HTML5 suporta, de raiz, conteúdos multimédia dinâmicos tais como o áudio e vídeo, dando mais liberdade às plataformas para utilizarem o *codec* de vídeo mais adequado.

Em consequência, a Google criou um formato de ficheiro multimédia designado por *WebM* que consistia em oferecer uma alternativa livre de royalties para utilizar na nova tecnologia HTML5. Consistia inicialmente num *container* com o *codec* VP8 para o fluxo de vídeo e o *codec* Vorbis para o fluxo de áudio. Isto lançou uma enorme competição dentro dos *codecs* de vídeo, onde por um lado o domínio do H.264 era praticamente completo, e por outro a Google oferecia uma tecnologia com resultados semelhantes e sem qualquer tipo de custo financeiro. O ITU-T em 2013 normalizou o *codec* de vídeo mais recente até à data, H.265, também conhecido por HEVC. Mais uma vez este viria a ter resultados superiores em relação ao seu antecessor, mas desta vez a política de *royalties* ficou mais agressiva fazendo com que muitas empresas do mercado *Web* que lidassem com o vídeo digital ficassem desapontadas com os valores pedidos.

Por coincidência nesse ano a Google lança o *codec* de vídeo VP9, que tal como o anterior, tem um desempenho idêntico ou até superior em alguns casos em relação ao seu concorrente direto (HEVC). Devido ao peso que a plataforma YouTube tem hoje em dia, esta plataforma serve de montra para que outras empresas tecnológicas analisem o funcionamento do VP9 e façam uma reflexão em relação à adoção eventual do HEVC. Em 2015, empresas como a Google e a Cisco estavam a desenvolver os *codecs* de vídeo VP10 e Thor respetivamente com o âmbito de obter melhores resultados em relação ao

[4] Royalty é um termo da língua inglesa que define o valor pago ao detentor de uma patente ou marca registada, de forma a permitir o seu uso e respetiva comercialização.
[5] HTML5 é uma linguagem de marcação para a navegação *Web* que fornece novas funcionalidades necessárias para aplicações *Web* modernas

HEVC e H.264 e sem custos para quem quisesse implementar a tecnologia. Mas em vez de continuarem a trabalhar de forma separada, juntamente com estas duas empresas foi criada uma aliança tecnológica sem fins lucrativos que reúne as outras empresas do setor incluindo as seguintes: Amazon, Apple, IBM, Microsoft e Mozilla. Esta Alliance for Open Media (AOMedia) tem como objetivo principal juntar forças e criar uma norma de codificação de vídeo digital para o futuro totalmente grátis, *open source* e dirigido para a Web. O projeto que surgiu para atender a essas necessidades foi o AV1, tratando-se da junção de elementos dos *codecs* Daala, Thor e VP10 criados pela Mozilla, Cisco e Google respetivamente. Atualmente o *codec* está em fase de testes, o YouTube disponibilizou uma lista com uma dezena de vídeos codificados em AV1 que ainda só são reproduzidos através de navegadores orientados para desenvolvedores. Ainda numa versão pouco otimizada, alguns testes já efetuados (Facebook, 2018) demonstraram de que o *codec* AV1 tem imenso potencial e será o principal candidato para ser a principal norma de codificação de vídeo digital na *Web* nos próximos anos, deixando o HEVC ao que tudo indica para um mercado mais virado para as transmissões televisivas.

Na atualidade, a norma H.264 tem-se deparado com dificuldades no mercado para se manter como a principal forma de codificação de conteúdos de vídeo digital. Com o aumento da qualidade e da resolução da imagem, que resulta da melhoria dos dispositivos de captura e reprodução de vídeo, esta norma não proporciona a solução mais eficiente para o armazenamento e a transmissão de vídeo digital. Na verdade, a indústria de redes de computadores aposta cada vez mais na melhoria das infraestruturas para proporcionar ao consumidor final a maior largura de banda possível. Uma das grandes motivações para esta aposta é precisamente a presença cada vez mais ubíqua de conteúdos multimédia com mais qualidade.

Como se pode observar na Figura 2.3, em 2017 (Público, 2017), a velocidade média em Portugal, na rede móvel, situava-se nos 19 Mbps para *download* e 8 Mbps para *upload* e, na rede fixa, 42 Mbps e 22 Mbps respetivamente. No que diz respeito à rede fixa, observam-se resultados bastantes satisfatórios, visto que o nosso país se situa em vigésimo-nono no ranking mundial de velocidades de acesso à Internet. No entanto, se nos concentrarmos nos conteúdos que nos rodeiam atualmente, é muito difícil transmitir uma *stream* de vídeo digital com uma resolução de 3840 × 2160 (4K), ou superior,

codificada em H.264. Por exemplo, para transmitir um conteúdo com essa resolução, a Netflix necessita de uma largura de banda que varia entre 15 a 20 Mbps (Netflix, n.d.).

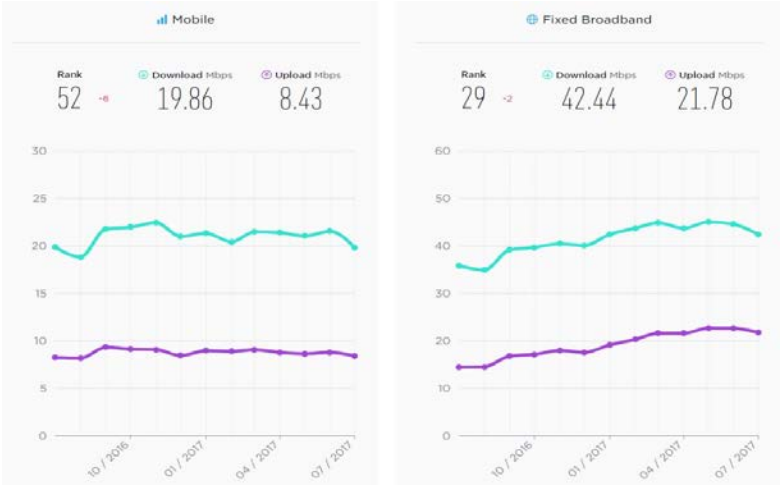

Figura 2.3 – Velocidade média em Portugal na rede móvel e fixa (Público, 2017)

2.2. Evolução da utilização de vídeo digital em difusão de sinal de TV

A televisão, tal como o processo de difusão de sinal de TV têm ambos uma história já longa, passando por constantes evoluções desde da era analógica até à digital. A primeira demonstração de uma transmissão televisiva (DAVIS, 2017) ocorreu no ano de 1926 em Londres na forma de um sistema TV eletromecânico produzindo uma imagem com cerca de 30 linhas, o suficiente na altura para pelo menos reconhecer uma face humana. Com o passar dos anos o mundo tem assistido a uma evolução muito rápida a nível tecnológico no processo de visualização do vídeo, desde a substituição da televisão mecânica para as televisões eletrónicas através dos *Cathode Ray Tubes* (CRT), até aos hoje muito utilizados *Liquid Crystal Displays* (LCD) e *Oganic Light-Emitting Diode* (OLED).

As transmissões televisivas a cores surgiram nos Estados Unidos da América em 1954 através da rede de televisão do país *National Broadcasting Company* (NBC). Foi um marco muito importante na história dos media porque até ter sido consolidada essa transmissão a cores, várias empresas da área da eletrónica estudaram a melhor forma de transmitir de forma analógica as cores, onde se destacou a RCA Corporation que fez com que fosse possível codificar de forma separada a informação do brilho e da cor, fazendo com que conservasse o máximo possível a largura de banda e mais importante do que isso, fazendo com que as televisões a preto e branco conseguissem exibir a imagem monocromática visto que a componente da saturação só interessava às novas

televisões a cores. Juntamente com esta evolução para a transmissão de sinal televisivo a cores surgiram três normas principais de codificação da cor no sinal analógico, nomeadamente e ordenada de forma cronológica, o NTSC, o PAL e o SECAM.

Em desenvolvimento há praticamente uma década, o NTSC recebeu a sigla e foi aprovado pela *National Television System Committee*, um comitê estabelecido pela comissão federal de comunicações norte-americano, de forma unânime semanas antes da primeira transmissão televisa a cores nos EUA. O sistema concebido tinha como características principais uma relação de aspeto de 4:3 (mais conhecido como *aspect ratio* em inglês), a transmissão de imagens com 525 linhas a uma taxa de varrimento de forma entrelaçada de cerca de 29.97 fotogramas por segundo, formados por 59,94 campos, em que cada fotograma consistia em dois campos de 262.5 linhas pares e ímpares. Este processo de entrelaçamento será explicado com maior detalhe mais adiante. Nessa altura não podiam aumentar a resolução da imagem sem incrementar a largura de banda, e como era necessário conseguir adicionar as cores sem aumentar a capacidade de transmissão do sinal e remover interferências entre o sinal a cores e o antigo, foi apresentada uma solução que consistia em sacrificar um pouco a taxa de fotogramas dos 30 (originalmente usados na transmissão a preto e branco) para menos três centésimos, além do mais de que tinha na altura o grande benefício de ser retrocompatível com as televisões antigas. O valor da taxa de fotogramas por segundo varia consoante a frequência da rede elétrica utilizada num determinado país, devido a certas questões como facilitar a conceção das televisões. Daí a norma NTSC ser adotada em todo o continente Norte Americano (exceto Gronelândia) e parte do continente Sul Americano por terem uma frequência de 60Hz na rede elétrica. Como em praticamente no resto do mundo esse valor da frequência é de 50Hz, outras normas tiveram de ser criadas devido à incompatibilidade com o valor da taxa de fotogramas utilizado no NTSC. Lado a lado podemos observar na Figura 2.4 que nos indica de forma clara a semelhança entre a frequência adotada na rede elétrica e as normas televisivas.

Com o objetivo de remover os problemas encontrados até à versão final da norma NTSC e de fornecer sinal televisivo a cores na Europa com uma frequência diferente da usada nos Estados Unidos da América e arredores, PAL e SECAM foram inaugurados no mesmo ano pela Alemanha e França respetivamente. Como as normas foram desenvolvidos de raiz para introduzir a cor nos ecrãs televisivos, não houve a

necessidade de arredondarem números, tendo assim ambos possuírem as mesmas características técnicas de 625 linhas apresentadas a uma taxa de 25 fotogramas por segundo (50 campos entrelaçados) numa relação de aspeto de 4:3 tal como o NTSC.

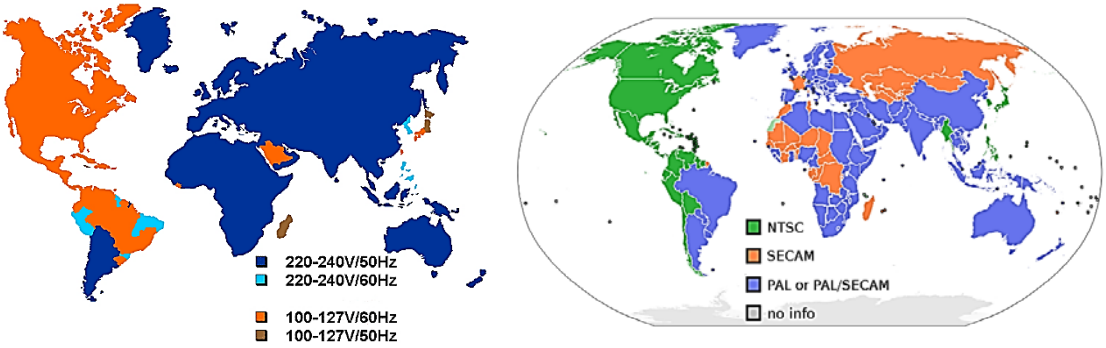

Figura 2.4 - Dados geográficos da frequência da rede elétrica e normas televisivas pelo mundo

2.2.1. Televisão Digital Terrestre (TDT)

Tal como referido anteriormente, uma das grandes limitações da transmissão televisiva analógica era a largura de banda do canal que rondava entre os 5 e 8 Mhz dependendo da norma utilizada. A necessidade de haver mais conteúdo televisivo com maior qualidade de imagem e som fez com que mundialmente se repensasse a forma de transmitir o sinal de TV para as televisões, nascendo assim por volta dos anos 90 o maior avanço tecnológico a seguir à televisão a cores, o conceito de DTV, sigla inglesa utilizada para designar a televisão digital. Uma das grandes vantagens é, sem dúvida, a grande eficiência no uso do espetro eletromagnético em relação ao sinal analógico, podendo através da multiplexagem[6] dos sinais, transportar múltiplos canais TV na mesma largura de banda utilizada por um canal transmitido de forma analógica (Pires, 2011), tal como podemos observar na Figura 2.5.

Tal como na transmissão de TV através de sinal analógico e desta vez sem terem que estar dependentes da frequência da rede elétrica local, foram várias as normas adotadas por diversos países pelo mundo para enviar de forma organizada o sinal TV digital, entre elas o DVB (*Digital Video Broadcasting*) que é usado maioritariamente no continente europeu e com presença em alguns países na África e Ásia. As normas ATSC, ISDB e DTMB são normas semelhantes utilizadas noutros países, mas que, neste trabalho, não serão muito abordadas, tendo o DVB maior importância e também sendo a norma mais utilizada no mundo inteiro para transmissões televisivas digitais.

[6] Multiplexagem é a operação da combinação de sinais analógicos ou digitais num único sinal para a sua transmissão sobre um único canal.

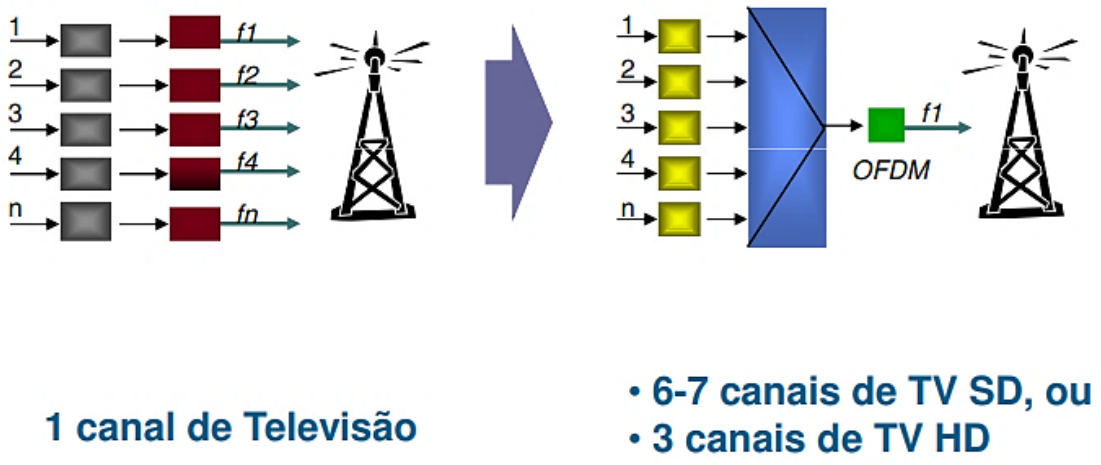

Figura 2.5 – Multiplexagem de canais TV

Resultado de um consórcio com mais de 270 empresas da área da transmissão televisiva com aprovação do comitê técnico EBU/CENELEC/ETSI, a norma DVB tinha como objetivos principais oferecer vantagens significativas em relação à transmissão analógica tendo sido propagada por mais de 80 países. Essa propagação hoje em dia é passível de ser feita por vários meios de transmissão existentes desde do transmissor até ao recetor. Como resultado foram criadas várias variantes do DVB para determinadas aplicações, das quais se destacam as seguintes:

- DVB-C (C-*Cable*)
    - Norma desenvolvida em 1994 para transmitir televisão digital sobre cabo, tipicamente do tipo coaxial. Teve como melhoria o DVB-C2 em 2008 com um aumento de eficiência no espectro de cerca de 30%.
- DVB-S (S-*Satellite*)
    - É possível transmitir o sinal TV através dos satélites que orbitam a Terra com esta norma desenvolvida em 1993. DVB--S2 foi uma versão melhorada em 2005 que coincidiu com a introdução do *codec* H.264, permitindo assim a transmissão de canais em alta definição.
- DVB-T (T- *Terrestrial*)
    - Dentro das variações DVB, esta é considerada uma das mais importantes por ter um número enorme de utilizadores e por ser bastante eficiente no

processo de transmissão. É através do ar, sem recorrer a satélites, que as transmissões são efetuadas com o método de multiplexagem OFDM[7]. Esta norma é utilizada em Portugal com a designação comercial TDT. Tal como nas duas variantes anteriores, esta também sofreu de uma melhoria intitulada de DVB-T2 que mais uma vez tem maior eficiência em relação ao seu antecessor, que neste caso até já introduz o novo algoritmo de codificação de vídeo digital (HEVC) em alguns países.

Apesar da televisão digital fornecer informações adicionais, tais como a listagem de programação dos canais, entre outras, estas variantes de transmissão de vídeo digital não têm à disposição certas características fundamentais hoje em dia, tais como a interatividade com o utilizador final, a possibilidade de organização e personalização dos canais segundo certas preferências do utilizador ou até a disponibilidade dos canais digitais em dispositivos móveis sem adaptadores dedicados à captação do sinal.

O avanço na tecnologia na área das redes de computadores permitiu com que houvesse um aumento substancial na largura de banda da internet para os utilizadores, dando justificações para que surgisse um novo método de entrega de conteúdo televisivo: o IPTV. Esta técnica permite o envio do sinal televisivo através do protocolo IP[8] como meio de transporte num túnel virtual que separa todo o tráfego comum da Internet do sinal de TV para garantir a qualidade do serviço. Uma característica interessante do IPTV é o facto de a comunicação ser feita de forma bidirecional criando uma interatividade entre o serviço de televisão e o consumidor final. São várias as possibilidades associadas a tal interação, desde o acesso a conteúdos e aplicações *online*, a possibilidade de se criar uma agenda de gravações de programas futuros, o acesso a listas de programas personalizadas por utilizador e, por fim, o VOD.

Do lado da empresa operadora de telecomunicações, esta tecnologia veio trazer uma forma de integração de vários serviços num só pacote: TV + Internet + Telefone, oferecendo custos reduzidos no mercado que permite ao consumidor final lidar de forma mais cómoda apenas com uma operadora. Dependendo da largura de banda

[7]OFDM (*Orthogonal Frequency Division Multiplex*) é uma forma de multiplexagem por divisão de frequências ortogonais. Neste caso, converte-se um fluxo de dados em série com uma taxa elevada de transmissão em vários fluxos em paralelo com uma taxa mais baixa de transmissão.
[8] IP (*Internet Protocol)* é um protocolo de comunicação que permite a transmissão de informação entre redes de computadores.

disponível, o IPTV permite enviar vários formatos de imagem. Através do ADSL a largura de banda consumida pelo IPTV é deduzida do total que é contratado. Por exemplo, quando um utilizador contrata o serviço de internet com uma velocidade de 12 Mbps, parte desse valor é repartido para a transmissão do sinal de TV. Dependendo dessa mesma velocidade é possível enviar determinados formatos de vídeo, sendo que a qualidade por norma ronda os 2 Mbps, em alta definição entre 6 a 8 Mbps e até aos 15 Mbps se for em ultra alta definição. Por questões de qualidade de serviço, muitas operadoras não permitem o envio de canais com qualidade acima da norma para não prejudicar o normal funcionamento da Internet no geral. No caso da fibra ótica é um pouco diferente, visto que a largura de banda utilizada para a transmissão do IPTV é partilhada por todos os utilizadores, não sendo assim contabilizada na largura de banda dedicada a cada cliente, daí essa tecnologia proporcionar uma margem melhorada para enviar canais com mais resolução da imagem.

A introdução da televisão digital trouxe maior qualidade de imagem aos televisores dos utilizadores, podendo transmitir diversos formatos dependendo da largura de banda disponível em cada tecnologia. Hoje em dia são utilizadas várias resoluções de imagem que se podem dividir em três principais por ordem de grandeza: *Standard-Definition* (SD), *High-Definition* (HD) e, mais recentemente, *Ultra-High-Definition* (UHD). Tendo por base as normas antigas de transmissão analógica (NTSC e PAL/SECAM), a definição padrão pode ser representada em duas resoluções distintas, 480i60 equivalente ao sistema norte-americano analógico das 525 linhas e 576i50 correspondendo aos sistemas PAL/SECAM de 625 linhas. O número associado a cada resolução indica o número de linhas visualizadas no ecrã da televisão, e as restantes podem ser utilizadas para transportar outras informações úteis tais como teletexto e sinais de teste, entre outros. Podendo tirar partido das vantagens da televisão digital, as resoluções de alta definição trouxeram mais qualidade de imagem numa relação de aspeto de 16:9, sendo esta a proporção mais utilizada na atualidade. Também conhecida por HDTV, esta é vastamente utilizada nas variantes DVB anteriormente abordadas tendo três principais formatos normalizados para a transmissão de TV digital definidos pelo ITU-R BT.709[9]: 720p (HD), 1080i e 1080p (*full HD*). O primeiro possui uma resolução de 1280 linhas verticais por 720 horizontais de forma progressiva, sendo que a designação do formato

[9] O ITU-R (*ITU Radiocommunication Sector*) é um dos três setores do ITU (*International Telecommunication Union*) responsável pela comunicação via rádio. BT.09 trata-se de uma norma aprovado em 1990 que indica o formato da televisão em alta definição, tendo um *aspect ratio* de 16:9.

está associada com o número de linhas horizontais tal como acontece em outros formatos já mencionados. Tal como sucede nos formatos SD , a taxa de fotogramas que é apresentado na TV é tipicamente dependente da região mundial que antes utilizava NTSC ou PAL/SECAM. Assim sendo, regiões como a Europa e Oceânia, entre outras, possuem imagens a 50 fps e nos outros casos a 60 fps. O mesmo acontece com o formato *Full HD*, com uma resolução de 1920 por 1080 pixéis, que pode ser transmitido de forma progressiva (*progressive* – 'p') ou entrelaçada (*interlaced* – 'i').

2.2.2. Entrelaçado versus progressivo

A técnica de entrelaçamento, inventada em 1930 pela empresa de telecomunicações Telefunken, foi bastante útil para combater certas limitações na transmissão do sinal de TV nesse momento. Como referido anteriormente, os televisores tinham que apresentar as imagens de forma sincronizada com a frequência da rede elétrica nacional (50 ou 60 Hz), sendo esta uma taxa muito elevada para apresentar os fotogramas com a largura de banda disponível e sem técnicas de codificação eficientes. A técnica do entrelaçamento consiste em enviar um sinal que contém dois campos de um fotograma capturados em tempos diferentes. Ou seja, o televisor nunca recebe nem disponibiliza no ecrã um fotograma completo. Ao invés, visualiza na horizontal as linhas pares do primeiro fotograma e nas linhas ímpares o fotograma seguinte sendo que cada segmento visualizado é a junção desses dois campos pares e ímpares, sendo que cada campo da imagem é exibido uma em cada 50 ou 60 vezes dependendo do sistema adotado (PAL/SECAM ou NTSC respetivamente). Podemos observar essa intercalação dos campos na Figura 2.6. Aproveitando o facto de que o nosso cérebro não consegue aperceber-se deste fenómeno de ilusão de ótica, este método permite assim visualizar imagens com maior resolução reduzindo assim a largura de banda necessária ao nível praticamente de um vídeo 720p, visto que o número de pixéis transmitidos é o mesmo.

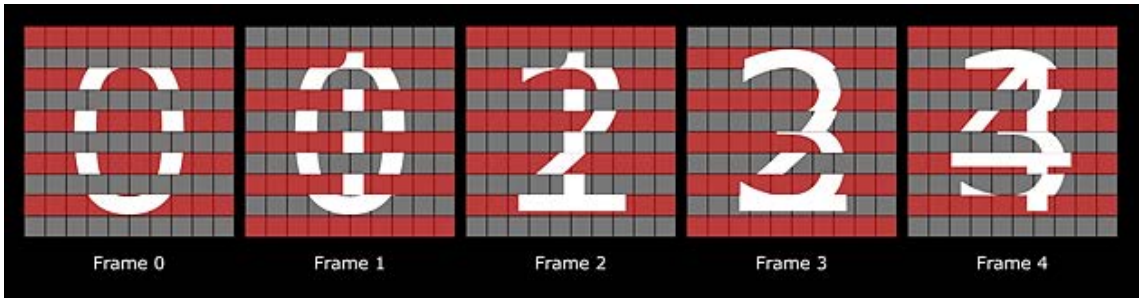

Figura 2.6 – Exemplificação de *frames* entrelaçadas

Esta técnica era muito bem-sucedida com ecrãs que exibiam de forma natural as imagens entrelaçadas, mas a indústria dos monitores dos computadores pessoais demonstrou que as imagens exibidas de forma progressiva apresentavam uma qualidade superior e mais nítida, fazendo com que o mercado dos televisores adotasse a mesma técnica até à atualidade. Isto criou um problema grande, ao exibir um vídeo entrelaçado num ecrã progressivo, este tem de atualizar uma imagem inteira 50 ou 60 vezes por segundo, ao contrário dos ecrãs entrelaçados que mostravam 1 campo (par ou ímpar) por segundo, mostrando os dois campos pares e ímpares de tempos diferentes ao mesmo tempo criando artefactos na imagem durante movimentos que ocorram no vídeo. Para permitir a visualização de um vídeo entrelaçado num ecrã progressivo foi desenvolvida uma técnica designada por desentrelaçamento (*deinterlacing*).

O principal objetivo deste método é apresentar um campo da imagem com linhas pares ou ímpares e preencher as restantes linhas através de algoritmos que calculam uma média apenas com a informação de um dos campos, ou guardam em memória os dois campos de imagens capturadas em momentos diferentes, para depois serem visualizados à velocidade da taxa de refrescamento do ecrã, fazendo com que seja uma transmissão com uma taxa de 50 campos por segundo ou de 25 fotogramas por segundo, sendo que essas anotações dependem da região em questão. Por exemplo, no caso europeu, o *European Broadcasting Union* (EBU) indica que as transmissões entrelaçadas em alta definição devem ser representadas com o número da resolução em questão, separado de uma barra diagonal com o número de fotogramas por segundo (1080i/25) (EBU-UER, 2010a).

Hoje em dia as televisões e as *TV Boxes* estão preparadas com os melhores métodos possíveis para desentrelaçar um vídeo de origem entrelaçada, mas ainda assim não é perfeito, principalmente em vídeos que contêm bastante movimento tal como são as transmissões de jogos desportivos. Imagens dispostas de forma progressiva no ecrã são uma mais-valia no sentido da qualidade do resultado final, apresentando todas as linhas da imagem de forma sequencial numa só passagem. Esta técnica tornou-se uma norma em todos os monitores de computador e nos televisores LCD e OLED. A EBU já afirmou numa declaração que as transmissões em alta definição (1080 linhas) seriam ideais ocorrerem de forma progressiva e não entrelaçada dada a evolução dos algoritmos de codificação de vídeo digital. Mas as principais estações televisivas não

concordaram com essas declarações por um motivo que é comum a todos: a escassez de largura de banda. Todas essas vantagens da transmissão do vídeo de forma progressiva vêm com um custo elevado no que toca à alocação de largura de banda necessária para transportar essa informação. Enquanto os formatos de vídeo 720p e 1080i apresentam débitos binários sem compressão por volta dos 0.5 Gbit/s, o formato 1080p mais do que duplica esse valor exigindo um débito binário de 1.2 Gbit/s (EBU-UER, 2010b), fazendo com que muitos, ainda hoje em dia, optem por transmitir canais de alta definição no formato 1080i, justificando-se que a tecnologia dos televisores está mais do que desenvolvida para desentrelaçar as imagens e apresentar uma resolução superior ao formato 720p e ocupar a mesma largura de banda.

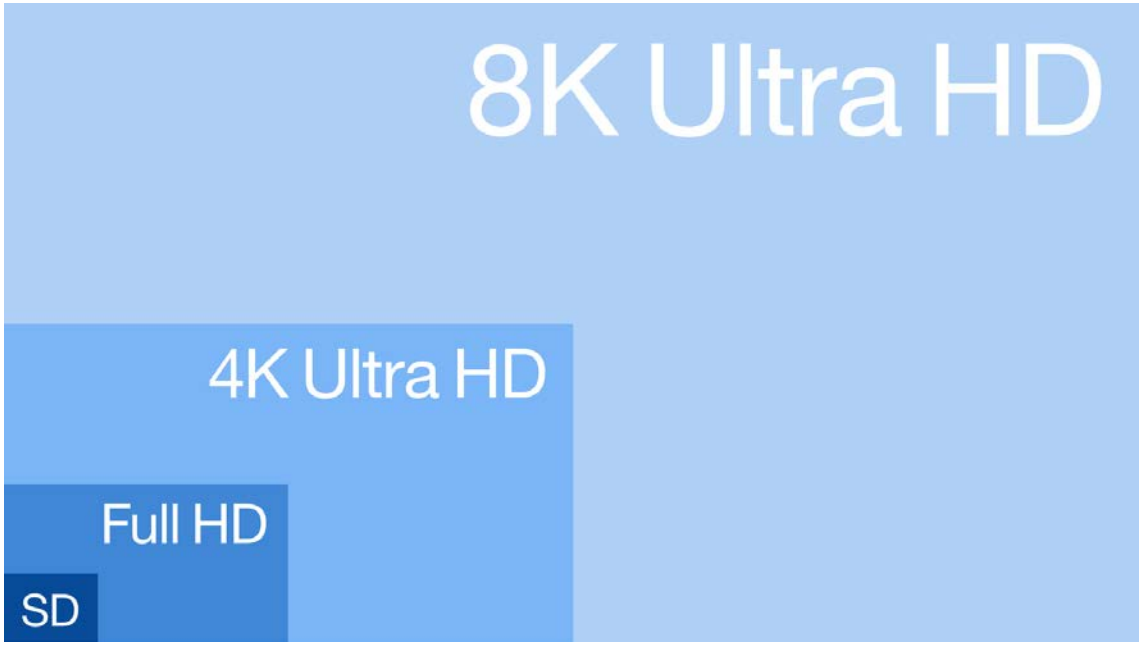

Figura 2.7 – Comparação entre os formatos SD, FHD e UHD.

A nível nacional todas as operadoras de televisão por cabo oferecem geralmente canais com formato de vídeo 576i e, quando há disponibilidade por parte da fonte do canal, também proporcionam canais em alta definição com a sigla HD junto ao nome do canal com o formato 1080i ou por vezes 720p nomeadamente no contexto desportivo. O único formato que não é transmitido garantidamente de forma entrelaçada e sim progressiva é o UHD (2160p), sendo que, devido à enorme utilização de largura de banda necessária, só é utilizado em ocasiões em que o contexto assim o justifique como jogos desportivos onde há questões económicas que favorecem o uso deste formato. Como podemos observar na Figura 2.7, o formato UHDTV apresenta duas resoluções que são

reconhecidas como 4K UHD (3840x2160 pixéis) e 8K UHD (7680x4320 pixéis), tendo 4 e 8 vezes mais resolução respetivamente do que o 1080p/i.

2.2.3. *Codecs* de vídeo na Televisão Digital

Os algoritmos de compressão de vídeo desempenham um papel crucial na transmissão do sinal TV de forma mais eficiente possível. Os primeiros *codecs* de vídeo digital foram pensados para áreas como a computação e videoconferência, entre outras aplicações. As estações televisivas estavam nessa altura muito céticas em relação ao uso de algoritmos de compressão que pudessem degradar a qualidade da imagem. O grupo de trabalho MPEG , na década de 90 do século XX, começou por pesquisar formas de codificar vídeo digital com muito movimento e que fosse possível armazenar em suportes físicos óticos tais como o CD-ROM. Entretanto, adotaram a primeira norma de codificação de vídeo digital (H.261) que obteve bons resultados a nível de rácio de compressão. Com a evolução positiva das transmissões televisivas, esse mesmo grupo criou uma versão chamada MPEG-1 que permitia finalmente a codificação de vídeo digital entrelaçado. As estações televisivas começaram a adotar o *codec* MPEG-2/H.262 que codificava vídeo entrelaçado com um débito binário entre os 4 e os 9 Mbps. Este algoritmo fez com que a televisão digital se expandisse pelo mundo inteiro com uma qualidade de imagem aceitável tendo possibilidade de codificar vídeos com resoluções até 1920 × 1080 pixéis com uma taxa de 30 fotogramas por segundo. Nos finais do ano de 1998 o *codec* começou a ser utilizado em Inglaterra através do canal televisivo BBC por transmissão digital terreste e pela Sky-Digital através de satélite.

Com o surgimento do formato HDTV, surgiram questões na Europa sobre a necessidade de criarem uma nova norma de codificação de vídeo digital para atender às necessidades dessas resoluções, criando-se assim o novo projeto intitulado de MPEG-3. Devido à grande versatilidade e estabilidade do MPEG-2/H.262 em atender às necessidades de grandes resoluções de vídeo, o MPEG-3 foi abandonado e deu margem para o antecessor crescer mundialmente nas transmissões televisivas. Em 2003 apareceu a primeira versão da norma MPEG-4/H.264 prometendo rácios de compressão 50% melhores em relação ao MPEG-2/H.262. Durante mais de 10 anos, esta técnica tornou-se uma norma fundamental nas transmissões televisivas de alta definição pelo mundo inteiro, sendo ainda, hoje em dia, utilizada em muitas transmissões. Na Europa são

muitos os países que optam por enviar os canais no formato SD codificados em MPEG-2/H.262 e canais HD com MPEG-4/H.264 em todas as variantes do DVB.

Em Portugal, todos os canais, independentemente do meio de transmissão (terrestre, satélite ou cabo), a codificação utilizada é MPEG-4/H.264 em qualquer formato de imagem, desde o SD até mesmo ao UHD. Tal como sucedeu com o seu antecessor, o *codec* MPEG-4/H.264 é bastante robusto e proporciona um bom desempenho em termos de eficiência da codificação de vídeo digital. Este *codec* apenas começa a exibir aspetos negativos quando se usa para codificar vídeos com 2160p ou 1080p sendo necessário uma elevada largura de banda para transmitir canais televisivos com essas resoluções. Segundo uma referência do ITU (Regula, 2016), programas emitidos em 720p/50 ou 1080i/50 necessitam em média de uma largura de banda a rondar os 6 Mbps já com áudio incluído e codificado em H.264. Se o vídeo for progressivo (1080p) pode ascender até aos 9 Mbps, o que faz com que muitas estações televisivas e operadoras de telecomunicações prefiram a transmissão do vídeo entrelaçado em alta definição (1080i). Para combater essas lacunas, em 2013, o mesmo grupo de trabalho responsável pelo desenvolvimento desse *codec* publicou formalmente o seu sucessor com o nome HEVC/MPEG-H Part 2. Com suporte até resoluções 8K UHD e com rácios de compressão superiores em relação ao seu antecessor, o HEVC começa a ser implementado em fases de teste atualmente por alguns países na Europa através da norma de transmissão terrestre DVB-T2. A mesma referência do ITU indica que existe uma redução em cerca de 50% no consumo de largura de banda em relação ao *codec* H.264. Alemanha (KRIEGER, n.d.), França (DVB.org, 2013) e República Checa (Thomson, n.d.) são alguns dos países que estão a testar este novo método de codificação para transmitirem formatos como 1080p/50 e até 2160p através da televisão terrestre, que antes era considerado impensável dada a largura de banda necessária para a transmissão.

Em paralelo, Portugal continua a transmitir os canais de forma terrestre na definição padrão (SD / 576i) codificados com H.264 sem previsões definitivas de melhorias no sentido de transmitirem formatos em alta definição, caso que apenas acontece em meios de transmissão como cabo/IPTV ou satélite. Muitas habitações ainda possuem uma ligação à Internet por ADSL (Online, 2017), o que implica que, em grande parte destes casos, não chegam sequer a dispor de 12 Mbps de largura de banda para *download*.

Nestes casos, as operadoras não conseguem transmitir canais em alta definição por IPTV, precisamente porque a transmissão do vídeo digital correspondente é codificada em H.264. Em média, para transmitir pela Internet um canal HD requer-se uma largura de banda que varia entre 6 a 8 Mbps. Por motivos que se prendem com políticas de qualidade de serviço, as operadoras são obrigadas a enviar os canais numa resolução menor para garantir a transmissão integral do conteúdo audiovisual. Ora, caso o sinal fosse difundido em HEVC, o *bitrate* necessário seria reduzido para metade daquele que é requerido atualmente para a codificação H.264. Isto significa que a difusão de um canal HD em IPTV alocaria, no mínimo, uma largura de banda de 3 Mbps. A Figura 2.8 apresenta a evolução dos débitos binários de *codecs* utilizados nas transmissões televisivas.

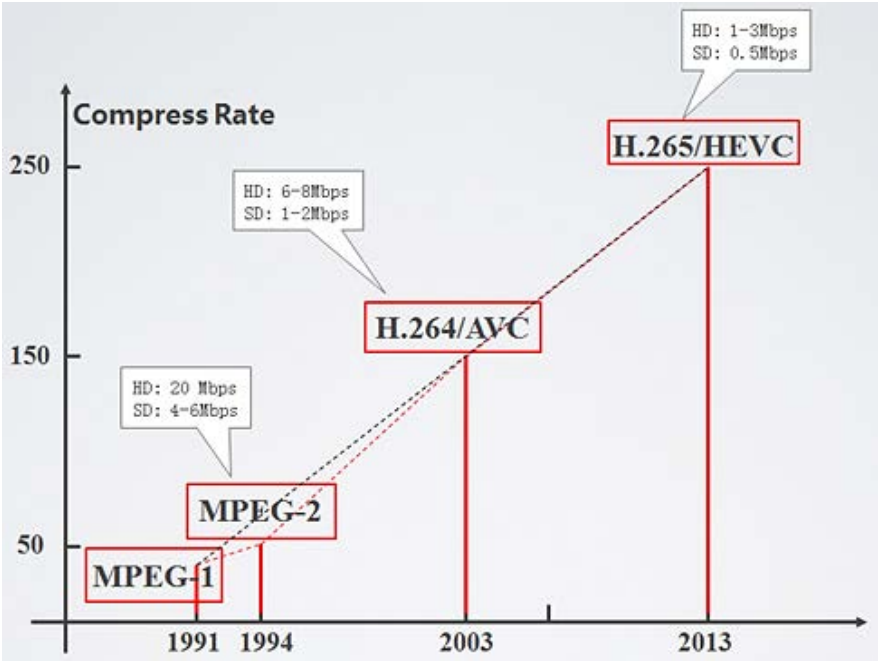

Figura 2.8 - Evolução dos *bitrates* dos *codecs* de vídeo utilizados nas transmissões televisivas

2.3. Compressão de vídeo digital: princípios de funcionamento dos *codecs*

O vídeo digital consiste num conjunto de imagens reproduzidas de forma sequencial a uma determinada velocidade. O olho humano consegue receber e analisar entre 10 a 12 imagens por segundo para ativar um mecanismo no cérebro que cria a sensação de continuidade visual, ou seja, a perceção de movimento (Paul Read, 2000). Cada imagem é composta por um conjunto de pixéis que, por sua vez, representam determinadas cores. Por exemplo, no caso do sistema RGB, qualquer cor se pode obter a partir da mistura de intensidade das 3 cores primárias: vermelho, verde e azul. Tipicamente, cada

componente da cor pode ser representada a nível computacional através de um determinado número de bits que indica a profundidade de cor da imagem. A nível digital, quando se quantificam todos esses parâmetros, é possível determinar o espaço de armazenamento ocupado por um ficheiro de vídeo digital num suporte magnético ou ótico. Por exemplo, dado um vídeo de alta definição com as seguintes características:

- Resolução: 1920 x 1080 pixéis
- Profundidade de cor: 8 bits (isto é, 8 bits por componente da cor)
- Taxa de fotogramas por segundo: 50 fps
- Duração: 1 min (60 s)

Multiplicando a resolução pela profundidade de cor obtém-se o espaço ocupado por aquela imagem (ou fotograma) naquele determinado instante. O produto desse resultado com a taxa de fotogramas por segundo a que é reproduzida a sequência de imagens representa, em termos práticos, o débito binário (ou *bitrate*). Multiplicando este último valor pelos 60 segundos obtém-se o valor total ocupado por esta sequência de vídeo, neste caso, corresponde a cerca de 17.38 *Gigabytes*. Este exemplo permite demonstrar claramente qual seria a dificuldade de armazenamento ou transmissão de vídeo digital se não existisse qualquer tipo de compressão de dados (Ribeiro&Torres, 2009).

Com a constante melhoria da qualidade do vídeo ao longo dos anos, tanto em termos do aumento da resolução como em termos do aumento da profundidade de cor, é cada vez maior a quantidade de informação digital com que os sistemas multimédia são obrigados a lidar, sendo, por isso, necessário que os vídeos sejam comprimidos da forma mais eficiente possível. De acordo com Ribeiro&Torres (2009), a palavra *codec* provém da associação de duas palavras inglesas: *coder*/*decoder*, tratando-se do conjunto dos vários algoritmos de compressão e descompressão de um determinado tipo de sinal digital podendo ser implementado em *hardware* e/ou *software*. Normalmente, é possível categorizar os *codecs* de duas formas distintas atendendo ao modo de compressão que aplicam: *codecs* sem perdas (*lossless*) e *codecs* com perdas (*lossy*). O objetivo principal de codificar um determinado tipo de *media* sem perdas da informação é o de permitir recuperar integralmente os dados originais através da sua reconstrução exata a partir da descodificação dos dados comprimidos. Pelo contrário, os *codecs* com perdas não

permitem recuperar exatamente a informação original. Contudo, os dados reconstruídos após a descodificação são suficientemente semelhantes de forma a serem úteis (normalmente a codificação apenas elimina dados que não são percetíveis aos sentidos dos seres humanos) de forma a ocuparem um espaço de armazenamento muito inferior e a requererem uma largura de banda de forma muito menor, tornando os processos de armazenamento e de transmissão dos dados comprimidos o mais eficientes possível. Geralmente, para obter melhores rácios de compressão, os algoritmos de compressão exploram as limitações percetivas do ser humano para reduzir ao mínimo possível a informação que permanece no ficheiro comprimido e que é estritamente necessária, quer para o ouvido humano (no caso da compressão de áudio digital), quer para o sistema visual humano (no caso da compressão de imagens *bitmap* estáticas ou da compressão de sequências de vídeo digital, que é precisamente o problema analisado no contexto deste trabalho).

A compressão de um certo tipo de media tem como objetivo principal a redução da sua dimensão em *Bytes* com uma perda mínima de informação. Por exemplo, no caso do vídeo digital isto consegue-se detetando semelhanças que possam existir entre *frames* sucessivas. Para a compressão de imagens pode encontrar-se um conjunto de técnicas de compressão concebidas para encontrar redundâncias, tanto a nível espacial como de cor. Tipicamente há sempre um conjunto de pixéis vizinhos que são semelhantes numa determinada zona da imagem. As normas de compressão de imagens, como é o caso do JPEG (*Joint Photographic Experts Group*), utilizam algoritmos que tiram partido dessas características para obter uma representação mais comprimida e, portanto, uma versão mais eficiente do ponto de vista da representação espacial. No que diz respeito às cores, realiza-se uma alteração do esquema de cores de forma a tirar partido de características naturais do sistema visual humano. Dado que, geralmente, o olho humano é muito mais sensível na deteção de alterações de caráter luminoso do que de mudanças cromáticas, em algoritmos de compressão de imagens e vídeo o espaço de cores RGB é convertido num sistema ligeiramente diferente designado por YCbCr que permite quantificar a componente responsável pela luminosidade (‘Y’) e também as componentes cromáticas azul e vermelho (‘Cb’ e ‘Cr’ respetivamente) (Ribeiro&Torres, 2009). A Figura 2.9 ilustra precisamente o momento em que um dispositivo fotográfico capta as 3 componentes RGB da imagem e as transforma para o sistema de cores YCbCr.

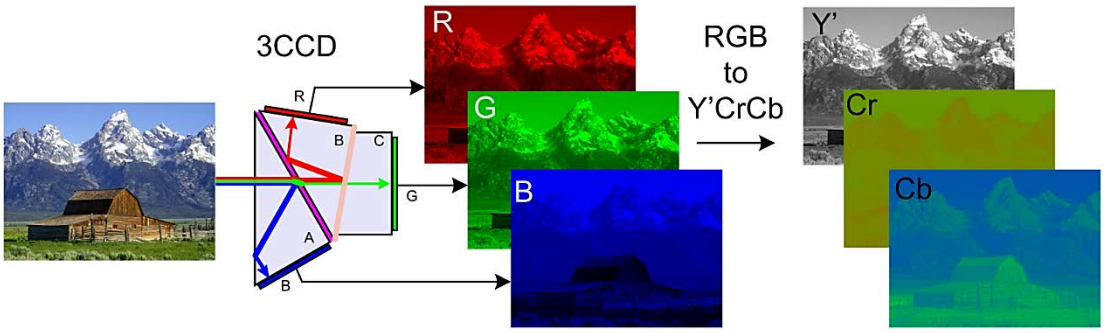

Figura 2.9 – Conversão de uma imagem RGB para Y'CrCb (LionDoc, n.d.)

Dado que um vídeo é constituído por uma sequência de várias imagens estáticas por segundo e não apenas uma, a compressão deste tipo de *media*, além de comprimir individualmente cada segmento de cada imagem, pode passar por encontrar semelhanças e diferenças entre as imagens que fazem parte dessa sequência. Cada uma das imagens é normalmente referida como fotograma (ou *frame*) como foi referido anteriormente. Por causa das características inerentes ao vídeo, numa sequência de *frames*, verifica-se que as imagens dessa sequência possuem praticamente o mesmo conteúdo visual mudando apenas pormenores pouco substanciais de *frame* para *frame* geralmente por causa do movimento. De facto, ao visualizar um vídeo que mostra, por exemplo, uma pessoa a caminhar, é possível observar que todo o fundo das *frames* que contém essa pessoa permanece praticamente inalterado se considerarmos que a câmara está fixa, sendo que apenas existe alteração de cores nos pixéis correspondentes às zonas onde a pessoa se desloca.

Por existir uma certa continuidade de imagem para imagem, foram desenvolvidos algoritmos para estimar previsões de movimentos que possam ocorrer nas imagens seguintes da sequência de vídeo, minimizando certas redundâncias que possam ser encontradas. A forma de eliminar ou reduzir muita informação redundante passa por guardar apenas as diferenças encontradas ao invés de armazenar dados redundantes (que ocupariam muito espaço adicional desnecessário). No caso do vídeo digital isto pode ser conseguido através da análise de redundâncias entre *frames* diferentes, por exemplo recorrendo a algoritmos de deteção de movimentos, e também com a análise da própria imagem para se detetar a existência de semelhanças. Estas duas abordagens à compressão de vídeo digital designam-se, respetivamente, por *Inter-Prediction* e *Intra-Prediction*.

Inter-prediction refere-se a um processo de compressão de vídeo para tirar vantagens de possíveis redundâncias a nível temporal devido à análise ser feita entre *frames* sequenciais, o que explica o uso do termo "*Inter*". Após ser feita a divisão da imagem em blocos de pixéis, esta técnica utiliza informação proveniente da *frame* anteriormente codificada, geralmente designada *por reference frame*, para encontrar semelhanças e apenas guardar num vetor (*Motion Vector*) a informação das posições dos blocos semelhantes e das respetivas informações. Mas isto sucede apenas em cenários nos quais a imagem permanece estável por um certo período de tempo, porque geralmente um vídeo tem movimento, tanto dos objetos como da própria câmara que está a recolher as imagens. Nestes casos o codificador trata de calcular as diferenças entre os blocos onde ocorreram esses movimentos e, antes de ser enviado para o descodificador, efetua-se a transformação desses valores residuais ou erros (também designados por diferenças). A compressão dos dados neste caso é conseguida devido ao envio da informação referente aos movimentos ocorridos entre *frames* através dos dados armazenados no MV (*Motion Vector*) e os respetivos erros de previsão, ao invés de enviar imagens completas que, muita das vezes, são muito idênticas e estar-se-ia a transmitir informação redundante. Assim feito, o descodificador consegue reconstruir a próxima *frame* com a receção da imagem referenciada do lado do descodificador e com a informação dos vetores e os erros associados. Mas este processo não pode ser completado apenas com o processo de compressão a nível temporal nos casos em que exista bastante ruído ou muito movimento na imagem. Nestes casos, poderá acontecer que a soma total do vetor de movimento com os respetivos erros possua uma dimensão muito maior do que a dimensão do bloco original, não havendo, portanto, compressão dos blocos. Outro problema consistiria na compressão de uma *frame* que foi previamente comprimida através do mesmo método, podendo propagar-se erros que conduzam à dessincronização do vídeo devido ao facto de não existirem imagens de referência. Estes motivos justificam que se utilizem segmentos de referência para tornar esta técnica útil e mais eficiente (Ribeiro&Torres, 2009).

Já a *Intra Prediction* é um método de compressão a nível espacial que pode ser usado em qualquer tipo de *frame*. Este método trata de aplicar algoritmos de previsão relativos à informação contida numa determinada imagem sem recorrer a *frames* vizinhas, o que justifica a utilização do termo "*Intra*". Este mecanismo, fundamentalmente, explora redundâncias espaciais que possam ocorrer dentro de uma imagem. Tipicamente,

objetos estáticos ou com pouco movimento, podem ser codificados através da diferença entre o bloco em questão e um outro bloco previamente codificado através da mesma técnica (conhecidos como blocos residuais). Quanto maior for a informação redundante disponível, maior será a capacidade de compressão proporcionada por esta técnica. Dependendo do algoritmo de compressão, existem vários modos ou técnicas predefinidas que calculam a área de um bloco com a informação proveniente dos blocos adjacentes, prevendo, deste modo, o resultado final apenas com a informação interna da *frame*. Alguns desses modos são inclusivamente utilizados por normas de compressão de imagens como é o caso do modo DC (*Distinct Mode*) que é idêntico ao que é utilizado na norma PNG, onde se utilizam pixéis adjacentes para o resultado final ser calculado soba forma da média entre os respetivos valores. Os modos aqui referidos são analisados de forma mais detalhada mais adiante nas subsecções referentes a cada *codec*.

Em relação às *frames* utilizadas nos métodos de previsão anteriormente descritos, podem ser definidas em 3 categorias distintas:

- I-Frame (*Intra-coded frame*):
    - Tal como a designação sugere, estas *frames* são codificadas internamente através do método de *Intra-Prediction*. Geralmente são muito necessárias do lado do descodificador para sincronizar a reprodução do vídeo no leitor em situações de pausa, *fast forward*, *rewind* ou, simplesmente, para a recuperação de possíveis erros. Estas *frames* tendem a ser menos eficientes do que as *frames* que usam *Inter-Prediction* mas, apesar disso, são muito úteis para fornecer informação fidedigna às *frames* seguintes para a correção de erros. Tal como foi referido anteriormente, o nível de compressão destes fotogramas depende muito da correlação possível na própria imagem: objetos, tais como fundos ou superfícies com cores uniformes tendem a ter um melhor resultado.
- B-Frame (*Bidirectional predicted frame*)
    - Tal como as P-Frame, este tipo de segmento recorre às mesmas técnicas de compressão, sendo que a diferença está na direção que podem tomar. De facto, neste caso é possível fazer uma previsão com base na *frame*

anterior tal como na posterior (trata-se de um processo bidirecional), tornando-se o tipo de fotograma que, por vezes, poderá necessitar de menos bits do que a P-Frame, e ao mesmo tempo a que necessita de maior poder computacional, mas que proporciona a melhor eficiência de compressão. Normalmente, estas *frames* são utilizadas entre as I-frame e as P-frame que servem de referência para armazenar os dados que são estritamente necessários. Na Figura 2.10 pode observar-se uma sequência de *frames* que exemplifica de forma simples quais os objetos guardados pelos 3 tipos de segmentos, ou *frames*, aqui descritos.

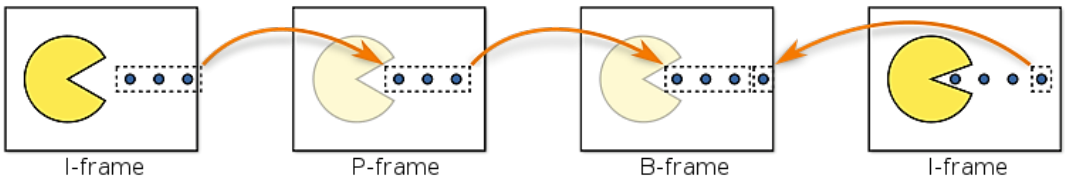

Figura 2.10 – Sequência de vídeo que representa as diferenças entre tipos de *frame*

Geralmente as *frames* que acabaram de ser referidas fazem parte de um grupo de imagens, também designado por *Group of Pictures* (GOP), como se pode observar na Figura 2.11, que especifica a ordem e a estrutura com que estas estão organizadas. Na realidade, um segmento de vídeo digital codificado consiste essencialmente numa sucessão destes grupos, fazendo com que o descodificador apenas necessite das I-frame para descodificar as imagens, ou *frames*, que se seguem. Estas podem ter formatos diferentes consoante o *codec* utilizado, mas geralmente iniciam-se com uma I-Frame como ponto de referência como início do GOP, seguido das P e das B *frame*.

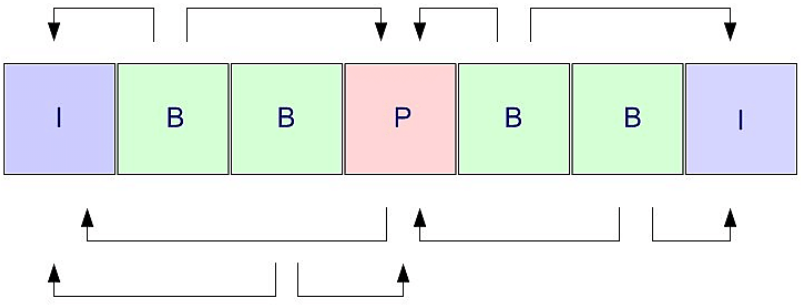

Figura 2.11 – Exemplo de um *Group of Pictures* utilizado no *codec* H.264 (Dalal & Juneja, 2018)

Na generalidade dos *codecs* de vídeo cada *frame* é dividida em vários blocos de pixéis de tamanhos variáveis consoante o algoritmo utilizado, sendo analisados pelo

codificador os elementos a serem tratados que possuam mudanças significativas. No caso dos *codecs* da família MPEG, estes elementos designam-se por *macroblock* (até ao *codec* H.264) e *Coding Tree Block* (CTB) no caso do HEVC. No VP9 e AV1 são conhecidos como *SuperBlock* (SP). Estes blocos são abordados com mais detalhe nas subsecções desta dissertação correspondentes a cada *codec*.

Após se efetuarem as previsões, o primeiro passo que se segue é a aplicação de transformadas à informação obtida pelas previsões, como por exemplo os vetores de movimento e os modos de previsão utilizados previamente no *Intra-coding* que reorganiza os dados de bloco para bloco em frequências de forma a reduzir redundâncias espaciais ou certas correlações que possam estar presentes em alguns resíduos. De uma forma geral, de acordo com Ribeiro&Torres (2009) a técnica mais utilizada nestes tipos de compressão é a *Discrete Cosine Transform* (DCT) que é composta por vetores ortonormais que produzem coeficientes das amostras. Apesar de nesta fase ainda não ocorrer compressão devido a esta operação poder ser completamente reversível (sem qualquer perda de dados), o resultado da transformada produz dados estatisticamente mais concentrados, produzindo distribuições de probabilidades de ocorrência dos resíduos com mais picos que possuem menor entropia e que podem ser comprimidos com mais rapidez e com maior eficiência por um codificador de entropia sem perdas.

A compressão começa a concretizar-se na fase do quantificador, sendo este um processo que implica perdas de dados. Mais uma vez, para aproveitar certas limitações da visão humana, este método explora as variações na luminosidade em altas frequências. O processo passa por dividir os blocos transformados por uma constante definida em forma de matriz e arredondar os valores obtidos em números inteiros. Após a divisão, uma constante QP (*Quantization Parameter*) define, dentro da matriz, um valor mínimo, transformando todos os valores em zero caso sejam iguais ou inferiores à constante. Em suma, quanto menor for o valor da constante QP, maior é a qualidade da imagem, mas maior é a degradação da eficiência da compressão do vídeo. Este valor é muito importante numa solução de compromisso entre a velocidade do processo de compressão e a qualidade imagem.

Finalmente, o processo de compressão passa pela codificação de entropia. Trata-se de um passo final de otimização sem perdas da representação binária que aplica um

método de compressão de dados sem perdas independente das características do sinal a comprimir, tendo como o objetivo a criação de códigos sem prefixo e atribui-los aos símbolos de entrada, isto é, substitui vários segmentos de vídeo com um padrão específico por códigos que gastam menos bits. Os métodos mais utilizados neste tipo de codificação incluem a Codificação de *Huffman* e a Codificação Aritmética. Na Codificação de *Huffman* atribui-se um código de tamanho variável a um determinado símbolo dependendo da respetiva frequência de ocorrência na *stream* original, quanto maior a repetição menor extensão terá o código binário correspondente. Por outro lado, a codificação aritmética separa a informação recebida em vários componentes e substitui por códigos, fazendo com que a *stream* inteira seja codificada com um único valor numérico fracionário. Com resultados melhores ou, por vezes semelhantes à Codificação de *Huffman*, este método peca apenas pelo uso intensivo de poder computacional para atingir o resultado final. Por isso, alguns *codecs*, como é o caso do H.264 e do HEVC, empregam formas de codificação de entropia mais recentes tais como o *Context-adaptive variable-length coding* (CAVLC) e o *Context-adaptive binary arithmetic coding* (CABAC). Estes métodos mais recentes tiram partido do número elevado de zeros nos blocos após a aplicação das transformadas e da quantificação dividindo separadamente em duas tabelas os coeficientes nulos e os restantes não-nulos, que permanecem numa outra tabela dedicada.

Como as *frames* são divididas em vários blocos de tamanho variável para efetuar as operações necessárias, tais blocos podem, por vezes, sofrer deformações e descontinuidades em relação à imagem original, criando artefactos na imagem com um efeito de distorção. Tais efeitos podem acontecer após alguns erros de previsão tanto nos modelos de *Intra* e *Inter Prediction*. A solução passa por aplicar um filtro designado por *Deblocking Filter* que reduz tal distorção entre blocos, com o objetivo principal de suavizar as arestas nesses intervalos. Na Figura 2.12 pode observar-se os processos principais envolvidos no codificador e descodificador.

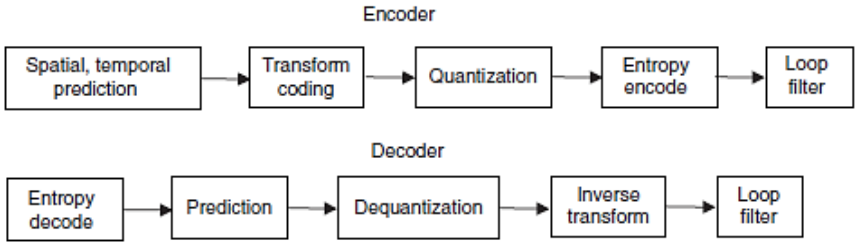

Figura 2.12 – Processo de codificação e descodificação de um vídeo digital (Troncy, Huet, & Schenk, 2011)

2.4. O *codec* H.264/AVC

O H.264 designa uma norma de compressão de vídeo digital desenvolvida por uma parceria denominada por *Joint Video Team* (JVT) que é constituída pelo grupo de trabalho da ITU (ITU-T *Video Coding Experts Group*) e pelo grupo *Moving Picture Expert Group* (MPEG). Também conhecido pelo nome comercial AVC ou MPEG-4 Part 10, este *codec* teve como grande objetivo proporcionar a mesma qualidade de vídeo em relação ao *codec* H.262 mas consumindo apenas metade do *bitrate*, sem aumentar de forma excessiva a complexidade para não dificultar a respetiva implementação. A primeira versão normalizada deste *codec* foi publicada a 30 de maio de 2003 (Union, 2003), sendo que, desde aí, é feita anualmente a publicação no ITU de novas versões.

É um *codec* ainda muito presente nos dias de hoje por causa da sua versatilidade e muito devido à respetiva capacidade de adaptação e escalabilidade para com os vários sistemas multimédia, elementos fundamentais para todo o tipo de aplicações em rede. Em relação a *codecs* anteriores, o H.264 oferece uma maior eficiência no processo de codificação, permitindo um menor uso de largura de banda na transmissão do sinal e também requer menos espaço para armazenamento do vídeo digital. Dados estes benefícios, o H.264 foi rapidamente adotado por imensas tecnologias relacionados com vídeo digital, desde dispositivos eletrónicos, normas de transmissão televisivas como é o caso do DVB e IPTV, até a aplicações de *streaming* na *Web* como é o caso da Netflix e do YouTube. A Netflix ainda utiliza atualmente o H.264 em resoluções até 1080p devido à sua baixa complexidade, boa qualidade de imagem e valores de *bitrate* aceitáveis, e apenas utiliza *codecs* mais recentes em resoluções 4K. No caso do YouTube, o H.264 apenas se utiliza *codec* no momento em que o vídeo é carregado para a plataforma e em transmissões em direto devido, mais uma vez, à baixa complexidade do algoritmo, permitindo que ambas as operações sejam feitas de forma rápida e no caso do live *stream* ter o menor atraso no tempo possível. Nesta plataforma. após o

carregamento do vídeo, converte-se posteriormente para o *codec* VP9 elaborado pela Google, que será analisado mais adiante neste capítulo.

Vídeos em alta ou baixa definição conseguem atingir um resultado aceitável em transmissões na rede de forma adaptável devido aos múltiplos perfis de codificação de que este *codec* dispõe. Os perfis neste e em outros *codecs* servem essencialmente para especificar quais os algoritmos a utilizar durante a codificação do vídeo. A escolha do perfil é efetuada consoante as capacidades de uma aplicação em concreto. Por exemplo, se a finalidade do vídeo passa pelo *streaming* na *Web*, terá um perfil em que minimiza a complexidade para aumentar a rapidez do processo de codificação. Ao mesmo tempo esta escolha permite essencialmente ao descodificador reconhecer em antemão quais os requisitos necessários para proceder à descodificação da *stream*. Os perfis do H.264 encontram-se divididos em três categorias principais; *baseline*, *extended* e *main*, e um adicional designado por *high*.

O perfil *baseline* é dirigido a aplicações com um ambiente propício a erros de transmissão, devendo minimizar o *bitrate* da *stream* e a sua complexidade, tendo por isso a menor eficiência de todos os perfis na codificação do vídeo digital. O *extended profile* é um perfil indicado para *streaming* devido a ser o único a usar *frames* adicionais na codificação (SI e SP *slices*) para facilitar a recuperação de erros que possam ocorrer durante uma transmissão. O perfil principal (*main profile*) permite transmitir melhor qualidade de imagem em relação aos dois perfis abordados. Porém, possui maior complexidade na codificação do vídeo devido à inclusão de B-frames e o algoritmo CABAC. Este perfil foi definido para as transmissões televisivas de sinal digital na norma DVB com resoluções padrão (SDTV). (Specification, 2009)

Os perfis *high* estão concebidos para implementar técnicas no sentido de transmitir a melhor qualidade possível não disponibilizando os mecanismos de verificação de erros de transmissão. O uso deste perfil justifica-se quando o vídeo digital foi gravado com alta qualidade de imagem (resoluções elevadas, 8 ou mais bits de profundidade, esquemas de cor 4:2:2 ou até 4:4:4). As principais aplicações deste tipo de perfil passam por transmissões televisivas em alta definição e armazenamento do vídeo digital com a maior qualidade possível. Os vários perfis *high* são diferenciados com nomes diferentes consoante o objetivo pretendido. Por exemplo, para codificar um vídeo digital de 10 bits de profundidade é usado o *High 10 Profile*. Juntamente com os perfis, o codificador

também permite estabelecer níveis (mais concretamente 16 níveis) que permitem especificar a resolução do vídeo em questão, o seu *bitrate* e o respetivo tamanho do *buffer*. Do lado do descodificador tanto o perfil escolhido como o nível devem ser suportados para que seja possível a reprodução do vídeo.

A norma H.264 reutiliza muitas técnicas que foram previamente adotadas pelas normas criadas anteriormente, mas possui funcionalidades inovadoras que a distingue das suas predecessoras e que permite obter melhores resultados e desempenho. Na Figura 2.13 é possível observar-se o diagrama representativo das diferentes fases do processo de codificação do vídeo digital definidas na norma H.264.

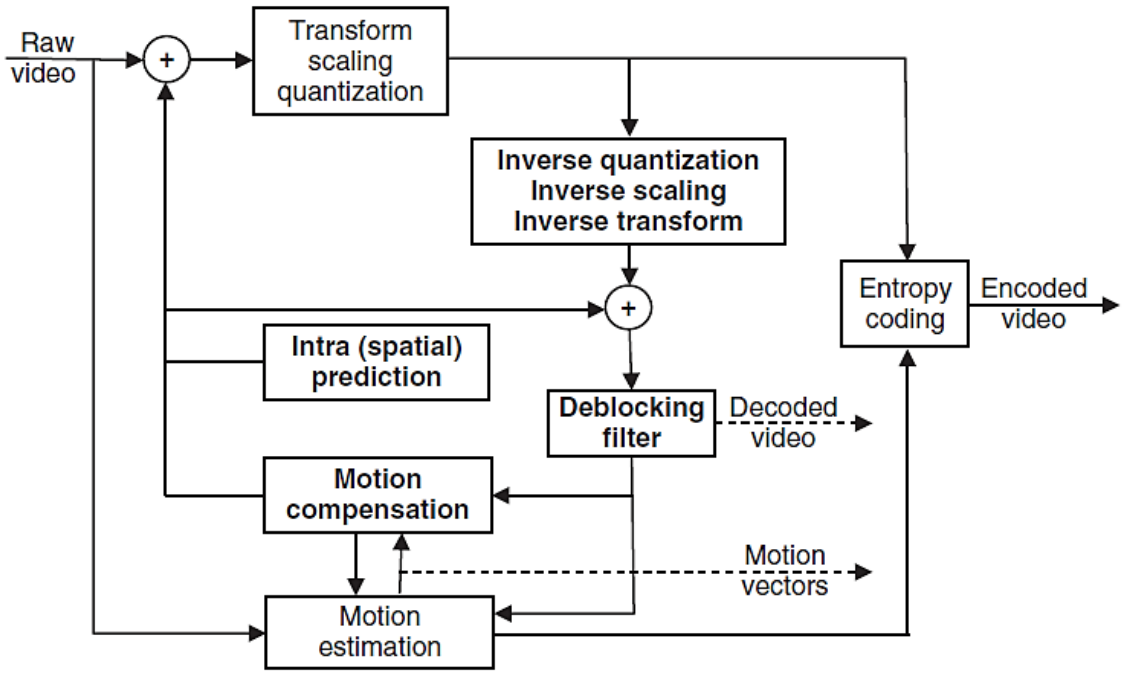

Figura 2.13 – Diagrama de blocos do codificador H.264 (Bing, 2015)

Na fase de *motion estimation* cada *frame* é dividida em blocos de tamanho 16×16 pixéis designados por *macroblocks*. Cada bloco poderá ser codificado utilizando blocos codificados internamente (*Intra frame coding*) ou utilizando blocos de *frames* passadas ou futuras para auxílio (*Inter frame coding*). Este processo de *motion estimation* tem como finalidade prever um conjunto de blocos no processo de *Inter frame coding*. Cada *macroblock* pode ser dividido em 3 dimensões predefinidas: 8×8, 16×8 e 8×16 pixéis. Os blocos de 8×8 podem ainda ser subdivididos em 4×4, 8×4 e, finalmente, 4×8. É importante referir que dimensão mínima dos blocos em *codecs* anteriores ao H.264 era de 8×8. Na realidade, blocos com dimensões mais reduzidas são muito úteis para

reduzir o efeito de *blocking artifacts*. Todas estas dimensões poderão ser utilizadas apenas no *Inter frame coding* (P e B *frames*). Na codificação espacial (*Intra prediction*) apenas são usados os blocos 4×4 e 16×16 para os diferentes modos de previsão. Nos blocos de dimensão 4×4 são 9 os modos que permitem calcular e prever internamente o valor da luminância de cada pixel. Excetuando o modo DC, que calcula a média dos pixéis vizinhos e preenche com esse resultado, os restantes modos representam a direção com que é atribuído o valor de cada píxel. A Figura 2.14 e a Figura 2.15 ilustram de forma clara os diferentes modos utilizados no *codec* H.264.

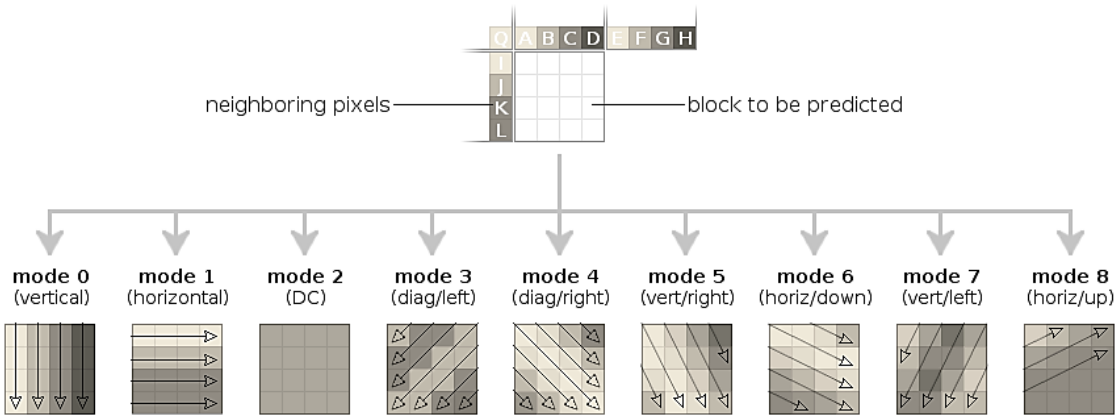

Figura 2.14 – Modos de *Intra Prediction* utilizados no H.264

Além destes 9 modos utilizados em blocos 4×4 no *Intra prediction coding*, existem 4 modos que podem ser aplicados em blocos 16×16, sendo que 4 são idênticos aos modos vertical, horizontal e DC abordados anteriormente e um modo adicional intitulado de *plane*. O *Intra Chrome Prediction* proporciona exatamente os mesmo modos 4 modos, mas são calculados em blocos 8×8. (Vanam, 2007)

Para além destas técnicas inovadoras que acabaram de ser referidas no setor do cálculo das previsões dos blocos, o uso de algoritmos modernos na fase de codificação de entropia, tais como como o CABAC e CAVLC, contribuiu igualmente para incrementar o rácio de compressão em relação aos *codecs* antecessores.

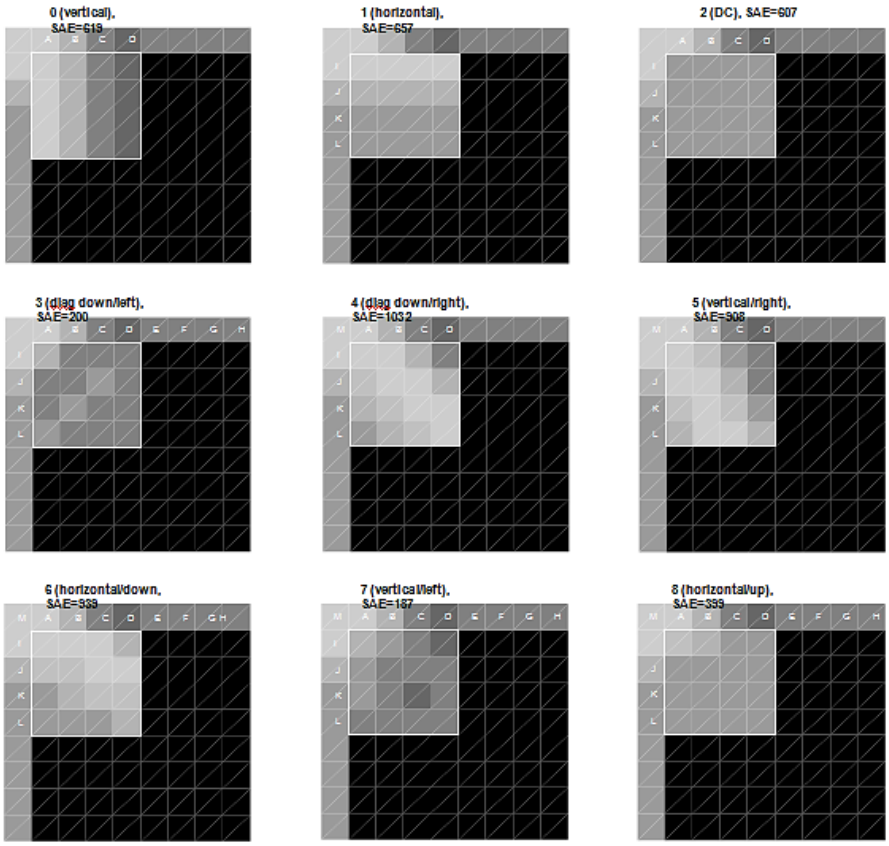

Figura 2.15 – Exemplificação dos modos de *Intra Prediction* do H.264

A norma H.264 poderá ter certos custos associados dependendo da sua finalidade. A organização MPEG LA gera as licenças associadas às patentes da norma. Esses pagamentos são realizados sob a forma de *royalties* nas vendas dos produtos que usufruem da tecnologia, tais como *TV boxes* e computadores, entre outros. Em agosto de 2010, essa organização anunciou que a codificação de vídeos para a Internet nunca seria alvo de pagamento de *royalties* desde que tais vídeos não tivessem custos para o utilizador final. (Reilly, 2015)

Finalmente, a designação deste *codec* é muitas vezes confundida com a designação x264, sendo esta última uma biblioteca de *software* que tem como objetivo implementar a norma H.264 para a codificação de vídeos. Desenvolvida pela organização *VideoLAN*, esta biblioteca é gratuita e é disponibilizada com uma licença *GNU GPL*[10].

[10] GNU GPL (GNU General Public License) é uma designação de licença para software geralmente livre, ou seja, sem custos para o utilizador, e de código aberto.

2.5. O *codec* VP9

O VP9 é um *codec* de vídeo digital *open-source* e livre de *royalties* desenvolvido pela *Google*, sendo que a primeira versão foi lançada nos finais do ano 2012 sob o projeto WebM. Com a evolução dos protocolos das redes e das capacidades computacionais dos dispositivos digitais, o vídeo digital na *Web* começou a ter cada vez mais impacto. Num momento que não existiam *codecs* completamente gratuitos, a Google iniciou o projeto WebM em 2010 juntamente com outras organizações como as *Xiph*, *Matroska* e *On2*, com o objetivo de desenvolver um formato de media para a *Web* completamente livre e gratuito para os utilizadores. A iniciativa WebM, além de identificar este projeto, é também a designação usada para identificar o formato de ficheiro (*file container*) que contém a *stream* de vídeo comprimida com o *codec* VP8 e a *stream* de áudio comprimida com o *codec* Vorbis. Com o incremento das solicitações de vídeo com resoluções mais elevadas na Internet, o VP8 deixaria de poder desempenhar o seu papel de forma eficiente, dando-se então prioridade ao desenvolvimento da nova geração do *codec*, em 2011, pela Google, intitulado VP9. Este projeto tem em mira o desenvolvimento de um *codec* capaz de produzir um débito binário muito inferior em relação ao antecessor, especialmente em *streams* de alta definição, sem incrementar muito o poder computacional exigido para a compressão do vídeo. Além das melhorias geracionais sobre o *codec* da mesma família, o VP9 também veio confirmar que seria capaz de proporcionar um melhor desempenho face ao H.264, podendo, por isso, ser uma alternativa ideal para a transmissão do vídeo digital na *Web* dado ser um *software* livre e gratuito.

Grande parte dos avanços tecnológicos incluídos no VP9 derivam de funcionalidades existentes no VP8. As características principais deste *codec* de vídeo passam por proporcionar uma maior eficiência na codificação, sobretudo para resoluções e profundidades de cor maiores.

2.5.1. *Prediction Block Sizes*

Na decomposição da imagem em blocos, o *codec* VP9 trabalha de forma muito similar ao H.264. A diferença é que nesta versão foi introduzido o conceito de *Superblock* que é semelhante ao conceito dos *macroblocks* do H.264, mas com uma dimensão máxima de 64×64 pixéis. Existem quatro modos de divisão do bloco neste *codec*: inteiro, na horizontal, na vertical e em quatro, sendo que na última opção poderá ser aplicado outra

vez esses modos de forma recursiva até tamanho mínimo de 4×4, como se pode observar na Figura 2.16.

O *codec* possui assim um total de 13 dimensões de blocos possíveis tendo em vista que os de maior dimensão são úteis e eficientes para codificar fragmentos em formatos de alta resolução como é o caso do UHD**UHD** (*Ultra High Definition*).

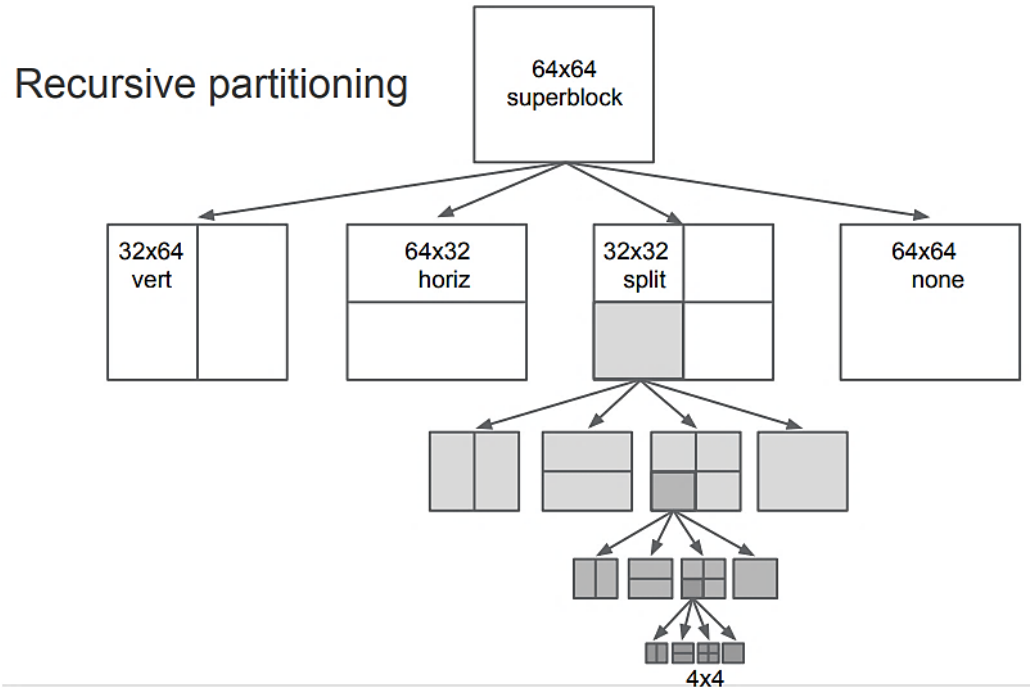

Figura 2.16 – Tamanhos e formatos dos blocos utilizados no VP9 (Mukherjee, Han, Bankoski, & S Bultje, 2015)

2.5.2. *Prediction Modes*

Tal como o H.264, o VP9 possui as mesmas designações para os modos de previsão espaciais e temporais: *Intra Mode* e *Inter Mode* respetivamente. No *Intra Mode* o VP9 possui um total de 10 modos de previsão organizados em blocos 4×4 até 32×32 pixéis. Os modos são semelhantes ao *codec* anteriormente analisado, contendo 8 modos direcionais, cada um com um ângulo diferente, que copiam o valor dos blocos adjacentes nessa direção, e dois que realizam operações aritméticas com os blocos vizinhos, sendo que um deles é comum ao que é usado no H.264 (DC). Caso a previsão espacial não seja aplicada, os blocos são calculados baseando-se em até 3 *frames* de referência criando vetores de movimento. São 4 os modos existentes no contexto desta técnica, organizados em blocos de 4×4 até ao limite máximo de 64×64: NEARESTMV, NEARMV, NEWMV e ZEROMV. O último modo significa que não há movimento e os

restantes modos geram uma nova lista de referências para o vetor baseando-se em informação de outras *frames*. O VP9 também pode fazer uso de *frames* invisíveis designadas por *Alternative Reference Frame* (ARF) que são utilizadas exclusivamente para servir de referência para a codificação de *frames* futuras que poderão estar desorganizadas, sem que esta seja visualizada.

2.5.3. Transformadas

No VP9 são utilizadas 3 tipos de transformadas: a DCT nos modos temporais com blocos 4×4 até *32×32*, *Asymmetric Discrete Sine Transform* (ADST) juntamente com DCT para os modos espaciais com blocos de 4×4 até 16×16 e a *Walsh-Hadamard Transform* (WHT) para a codificação sem perdas apenas em blocos 4×4, existindo um total de 14 combinações de transformadas possíveis com dimensões de um quadrado.

2.5.4. Codificação de entropia

No que toca à codificação de entropia, o VP9 utiliza a codificação aritmética ao invés de atualizar as probabilidades consoante a contagem dos símbolos durante a codificação ou descodificação, como acontece no CABAC, que é utilizado nos *codecs* H.264 e HEVC. No VP9 as probabilidades são atualizadas antes da codificação de próximas *frames* baseando-se na entropia dos dados de *frames* anteriores, e não durante. Assim, este método pode tirar partido da redundância espacial que pode ocorrer entre *frames*, mantendo as probabilidades constantes durante a codificação de cada imagem fornecendo assim uma compressão eficiente em *Inter frames*.

2.5.5. Técnicas de *Bitstream*

Além das técnicas de compressão acima descritas, o VP9 também fornece ferramentas concebidas para dar suporte a certas ocorrências que possam suceder durante a transmissão da *stream* de vídeo num ambiente adverso como é a Internet. Por utilizar um codificador aritmético, bits errados podem surgir numa *frame*, o que faz com que seja impossível a descodificação desse sinal nas *frames* subsequentes. Um dos métodos usados neste *codec* é o modo de resiliência de erros durante a codificação do sinal. O VP9 suporta a definição de uma variável *error_resilient_mode* que, quando está ativa, faz com que o contexto do codificador mude para que as *frames* deixem de depender entre elas e passem a utilizar técnicas para não propagar erros, tais como reiniciar o contexto de codificação no início de cada *frame*, ou vetores de movimento de referência que não podem usar valores de vetores utilizadores em imagens anteriores.

Estas restrições permitem um melhor desempenho na rede, mas conduzem a uma diminuição no desempenho em cerca de 4 a 5%. (Mukherjee et al., 2015)

2.5.6. *Tilling*

O VP9 suporta, quando disponível, a descodificação de múltiplas *frames* em paralelo e além disso, cada imagem pode ser subdivida em unidades, designadas por *Tiles*, que serão processadas por diferentes *threads* de forma independente, aumentando significativamente a velocidade na codificação ou descodificação na presença de um codificador com capacidade para *multi-threading*. (Mukherjee et al., 2015)

2.5.7. Implementações

O VP9 é implementado por *software* através da biblioteca *libvpx,* desenvolvida pela Google, que serve de referência não só a este *codec* como também ao VP8 e à fase inicial do AV1, até ser criada uma nova biblioteca baseada nesta e designada por *libaom*. A biblioteca *libvpx* é gratuita e tem código fonte livre.

Dado que o VP9 é desenvolvido pela Google, este *codec* é muito utilizado na plataforma de partilha de vídeos YouTube em toda a biblioteca de vídeos armazenados nos servidores em todas as resoluções que lá são apresentadas. Além do YouTube, a Netflix, outra plataforma de vídeos muito popular, também adotou uma versão do VP9 para dispositivos móveis na plataforma Android com o objetivo de diminuir a largura de banda necessária para visualizar o conteúdo e também reduzir o espaço ocupado em disco quando o utilizador decide transferir o conteúdo para o seu dispositivo (Andrey Norkin, Jan De Cock, Aditya Mavlankar, 2016).

Devido ao VP9 estar inserido em plataformas de vídeo muito populares na Internet, é importante que seja ser suportado pelos navegadores *Web* mais populares para permitir a reprodução fácil do vídeo digital. Foi do interesse do Google ser a primeira organização a adicionar esse suporte nos respetivos navegadores (Chromium e Google Chrome) em meados de 2013. O navegador Opera adicionou o suporte ao *codec* VP9 nos meses seguintes, seguido do Mozilla Firefox e do Microsoft Edge. Ainda no domínio dos navegadores *Web* mais populares na Internet, a Apple decidiu não implementar o VP9 no seu navegador *Web* Safari, preferindo adicionar o suporte do *codec* HEVC.

2.6. O *codec* H.265/HEVC

O HEVC é um *codec* de vídeo desenvolvido pelo mesmo grupo de trabalho que concebeu o *codec* H.264, sendo que o HEVC também é conhecido como H.265 por ter sido aprovado pelo ITU-T, sendo igualmente designado por MPEG-H *Part 2* devido ao envolvimento do grupo MPEG na normalização deste *codec* de vídeo digital para o setor das transmissões televisivas. Como é conhecido pelos seus antecessores, a cada década correspondente ao desenvolvimento de cada *codec* de vídeo, a eficiência da codificação do vídeo digital tem melhorado cerca de 50%, desde o H.264 em relação ao H.262, verificando-se o mesmo com o H.265 em relação ao H.264. Sendo considerado como uma extensão do trabalho desenvolvido no *codec* H.264, o HEVC é o resultado de um trabalho contínuo de mais de 10 anos, sendo que a primeira versão da especificação deste *codec* foi aprovada a abril de 2013. O H.264/AVC tinha um bom desempenho nas resoluções utilizadas nos anos em que o *codec* foi lançado, mas devido ao desenvolvimento das tecnologias multimédia, conteúdos visuais com mais definição foram emergindo ao longo do tempo, fazendo com que fosse necessário existir mais capacidade de armazenamento e de largura de banda para suportar tais conteúdos. O HEVC serve para o propósito de reduzir a o espaço requerido e a largura de banda necessária de um vídeo digital progressivo mantendo a respetiva qualidade e também para vídeos com formatos de ultra alta definição com resoluções até 7680×4320, retirando o suporte para a codificação de vídeos entrelaçados.

As subsecções que se seguem analisam alguns dos aspetos técnicos mais importantes que demonstram a maior eficiência do HEVC/H.265 em comparação com o AVC/H.264.

2.6.1. *Picture Partitioning*

O AVC/H.264 utiliza a terminologia *macroblock* para o conceito de estruturas de 16×16 pixéis numa imagem onde são realizados os cálculos durante o processo de codificação do vídeo. Este conceito é semelhante ao que se utiliza no HEVC que, ao invés de dividir uma imagem em múltiplos *macroblocks*, divide uma imagem em estruturas mais flexíveis designadas por CTB. Dependendo das definições introduzidas no codificador, os CTB podem adquirir dimensões como *64×64*, *32×32* e 16×16 pixéis, sendo que os CTB com as dimensões máximas são mais eficientes em vídeos com resoluções

elevadas, mas também demoram mais tempo a ser obtidos durante o processo de codificação (Ohm & Sullivan, 2013).

Cada CTB pode ser organizado em divisões recursivas numa estrutura do tipo *quad-tree* até às dimensões de 8×8 pixéis. Estas subdivisões de um CTB são conhecidas como *Coding Unit* (CU) e são responsáveis pela codificação das diferentes zonas da imagem, sendo que adquirem dimensões diferentes dependendo do nível de detalhe da imagem. Geralmente as CU com dimensões mais reduzidas são utilizados em zonas de uma imagem mais detalhadas como arestas e delimitações, ao passo que CU com dimensões superiores são utilizadas em zonas básicas sem detalhe. Cada CU pode ser tratada no processo de predição de imagens e nas transformadas, que serão abordados mais adiante. Na Figura 2.17 é demonstrado um exemplo de subdivisão de um CTB em várias CU do lado esquerdo e, do lado direito, observa-se a representação em árvore dessas CU, sendo que cada nível da árvore representa uma dimensão diferente.

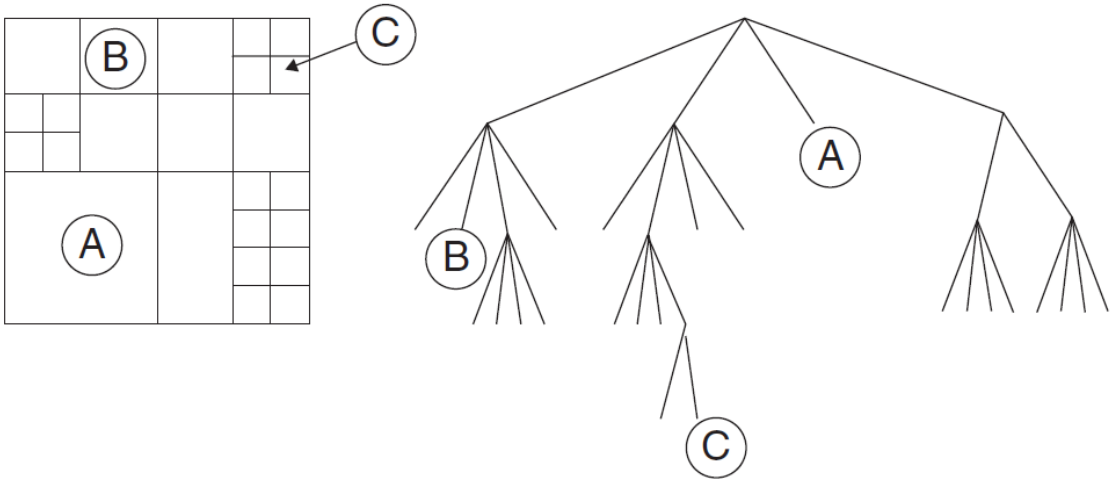

Figura 2.17 – Exemplo de um CTB *64×64*. (A) CU *32×32*, (B) 16×16, (C) CU 8×8. (W. Barz & A. Bassett, 2016)

2.6.2. Modos de Predição

Para além das *Coding Unit*, os CTB podem ainda ser subdivididos de forma homogénea em *Prediction Units* (PU), elementos utilizados nos diferentes modos de predição: *Intra-prediction* e *Inter-prediction*. Diferentes CU podem ser divididos até um total de 8 combinações possíveis de PU, tal como se pode observar na Figura 2.18, sendo que apenas as partições 2N×2N e N×N são utilizadas no *Intra-prediction* e restantes no *Inter-prediction*.

O modo *Intra-prediction* do HEVC funciona de forma similar ao AVC que foi revisto na secção 2.3. Além dos 9 modos de predição do AVC, o HEVC possuí até 35 modos diferentes de predição, sendo que 33 deles são angulares, e 2 (modo DC e Planar) derivam do AVC. Tal como no seu antecessor, o *Intra-prediction* permite prever informação trabalhada num determinado PU consoante os dados obtidos nos blocos vizinhos.

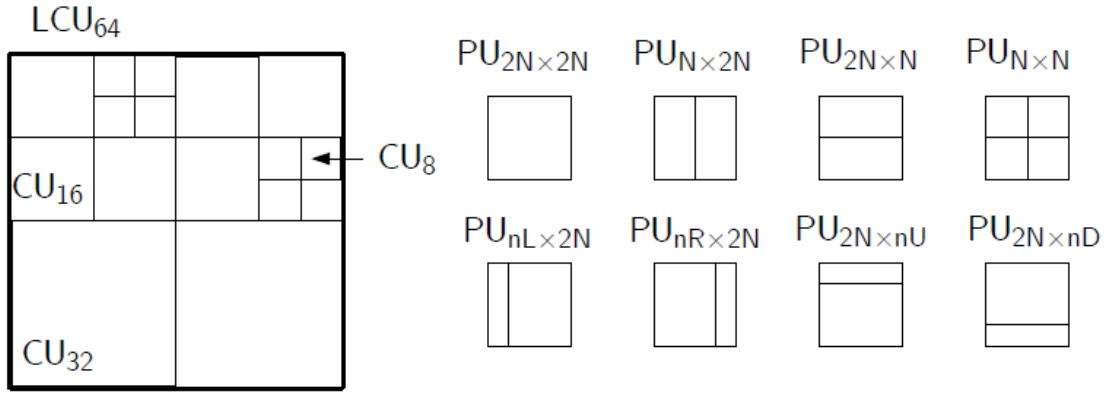

Figura 2.18 – Representação gráfica das combinações possíveis de *Prediction Units*. (C. Antoniou, 2017)

Como visto anteriormente, o *Inter-prediction* utiliza a informação dos pixéis obtidos em *frames* passadas e futuras como referência e permite guardar as diferenças obtidas em vetores de movimentos (MV) para reconstruir a informação de forma a comprimir os dados. A previsão de movimento no HEVC é mais complexa do que no seu antecessor, tendo dois modos principais: *Advanced Motion Vector* e *Merge Mode*. O *Advanced MV* cria uma lista de vetores de movimento candidatos utilizando lógica probabilística complexa recorrendo a dados temporais e espaciais, sendo que apenas os melhores candidatos são transmitidos na *stream* de vídeo. O *Merge Mode* é efetuado de forma similar apenas utilizando vetores vizinhos, equivalente ao *Skip Mode* utilizado no *codec* AVC.

2.6.3. Processamento Paralelo

Sendo que o processo de descodificação de *streams* de vídeo HEVC se tornou mais complexo em relação ao AVC devido à adição de novos elementos e técnicas, a descodificação do vídeo com recurso ao paralelismo tornou-se fundamental para tornar o processo mais veloz. Além da descodificação paralela em *slices* utilizada no AVC, o HEVC também permite tal processamento em *Tiles*. Os *Tiles* são constituídos por

componente retangulares de CTU (cada *Coding Tree Unit* consiste nas componentes de luminosidade dos CTB, juntamente com as componentes correspondentes das cores e outros elementos adicionais) que são descodificados de forma independente, sendo que cada *tile* é atribuído a um novo processo (*thread*), como se ilustra na Figura 2.19.

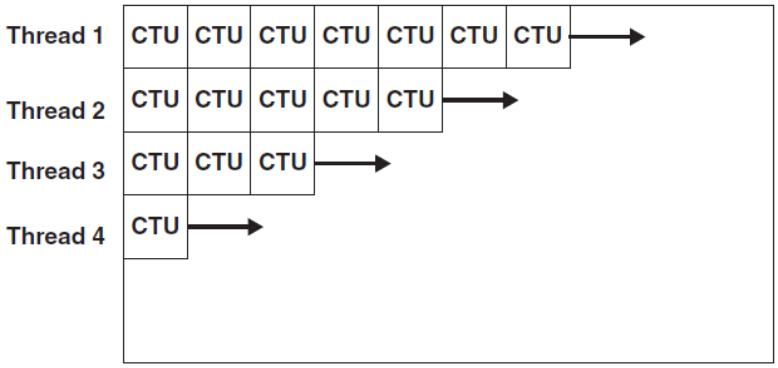

Figura 2.19 – Processamento paralelo no HEVC (W. Barz & A. Bassett, 2016)

2.6.4. Transformadas e Quantificação

A transformada de blocos, como visto anteriormente, é utilizada para codificar erros residuais (ou diferenças) resultantes das previsões temporais e espaciais (*intra* e *inter predicition*). Quanto maior as dimensões dos blocos utilizados neste processo em áreas homogéneas das imagens, maior serão os desempenhos obtidos na codificação. Como as resoluções de imagem utilizadas nos vídeos tendem a ser maiores ao longo do tempo, as sequências de vídeo em alta definição possuem mais partes com muita correlação espacial na imagem. Para tirar partido destas características, o HEVC introduz três novas dimensões de estruturas de transformadas (16×16, *32×32* e *64×64* pixéis) além das duas dimensões utilizadas no AVC (4×4 e 8×8 pixéis). Com o aumento destas dimensões, a complexidade deste processo é proporcional. O HEVC utiliza o algoritmo *fast DCT* para minimizar essa complexidade.

Após as operações das transformadas, os coeficientes resultantes são quantificados da mesma forma que o AVC e reorganizados num vetor unidimensional para a última fase de otimização operada pela codificação de entropia. Na codificação de entropia o HEVC utiliza apenas o algoritmo CABAC de forma idêntica ao AVC mas com pequenas modificações de modo a simplificar o processo de descodificação em paralelo.

2.6.5. Implementações

O HEVC tende a ser implementado nos mercados do setor televisivo no qual o MPEG tem ocupado essa posição ao longo dos últimos anos. Como se explica na secção 2.2, o HEVC foi integrado na norma de transmissão de televisão digital terrestre DVB-T2 de modo a dar continuidade à melhoria da qualidade de imagem e tornar as transmissões mais eficientes. Na *Web* a implementação deste *codec* de vídeo tem vindo a ser mais demorada, em grande parte devido às políticas de licenciamento e ao pagamento de *royalties* aplicado às empresas que pretendem incluir esta tecnologia nos seus produtos. Estas políticas são analisadas com maior detalhe a seguir.

Atualmente, o HEVC está inserido em três plataformas principais: televisão, dispositivos móveis (Apple iOS) e computadores pessoais. Como a Google desenvolve o sistema operativo Android para os dispositivos móveis, é natural que os dispositivos Android integrem os *codecs* de vídeo digital produzidos pela Google, como é o caso do VP9 e do AV1. A Apple tomou a decisão, em 2017, de adotar o HEVC como nova norma de codificação de vídeo digital para a plataforma iOS e MacOS, incluindo suporte em *hardware* nos *chips* gráficos e em *software* nas aplicações como é o caso do navegador *Web* Safari (Apple, 2018). Nos computadores pessoais, a aposta é feita pelos fabricantes principais de *chips* gráficos para os computadores pessoais (Intel, Nvidia e AMD) que pretendem incluir o suporte em *hardware* do HEVC para produtores de conteúdos multimédia e para os consumidores finais.

2.6.6. Licenciamento e *royalties*

Tal como o seu antecessor, o HEVC também se rege através de políticas de pagamentos de *royalties* nos produtos que integram esta nova norma de codificação de vídeo digital. Atualmente a *HEVC Advance LLC* é o grupo independente que gera as licenças e patentes referentes a esta tecnologia e que é responsável por disseminar a norma pelo mundo inteiro através da comunicação entre esta entidade e as empresas interessadas em implementar o HEVC nos respetivos produtos. Apesar deste processo de pagamento de *royalties* ser semelhante ao que ocorria com o *codec* AVC/H.264, os valores prestados na última revisão do documento realizado em 2018 são substancialmente mais elevados. Os fabricantes dos dispositivos móveis, entre eles *smartphones*, *tablets* e computadores portáteis, estão atualmente sujeitos ao pagamento de 0.40$ por cada dispositivo vendido com o *codec* de vídeo implementado em *hardware*. Fabricantes de

produtos como câmaras de videovigilância, *TV boxes* e leitores de DVD/Blu-ray pagam entre 0.20$ a 0.80$ por cada dispositivo dependendo do respetivo custo de venda ao consumidor final. Os dispositivos em que o pagamento de *royalties* é maior são as televisões com resoluções UHD (4K), em que o fabricante paga 1.20$ à *HEVC Advance LLC* por cada dispositivo vendido independentemente do custo de venda final (HEVC Advance LLC, 2018). Para aliviar o pagamento destas taxas, Peter Moller (CEO da HEVC Advance) anunciou que dispositivos que não dispusessem capacidades para codificar ou descodificar conteúdos HEVC em *hardware* não estariam sujeitos ao pagamento de quaisquer taxas desde que o processo fosse conduzido por *software* (HEVC Advance LLC, 2016).

O pagamento de taxas pela utilização do HEVC é um dos principais motivos pelos quais a adoção desta nova norma de vídeo ser ainda muito baixa, principalmente no setor da Internet. Um grande número de empresas tecnológicas, incluindo, entre outras, a Amazon, a Google e a Intel, formaram uma aliança (AOM) e estão a criar uma nova norma de codificação de vídeo digital completamente gratuita e sem o pagamento de quaisquer taxas designada por AV1.

2.7. O *codec* AV1

O simples facto de se conseguir visualizar vídeos em qualquer lugar e em qualquer momento, com uma resolução de imagem maior, e com mais *frames* por segundo, levou a um incremento na transmissão do vídeo digital na Internet em relação à transmissão tradicional por difusão, sendo cada vez mais importante que essa transmissão seja comprimida de forma eficiente. Como foi indicado anteriormente, o H.265/HEVC proporciona desempenhos muito superiores sobre o seu predecessor H.264/AVC, mas, inclui políticas de pagamentos de *royalties* severas. Em meados de 2013, na mesma altura em que se efetuou a normalização do HEVC, a Google lançou no mercado o *codec* de vídeo VP9 que viria a ser um candidato bastante competitivo e completamente livre de quaisquer custos. Devido aos bons resultados obtidos por este *codec* a nível de eficiência, e à sua rápida adoção na *Web*, muito devido à respetiva implementação no YouTube e na Netflix, gerou-se um grande interesse no desenvolvimento de uma próxima geração deste *codec* no sentido de continuar com o trabalho desenvolvido até ao presente, de modo a focar a nova versão na entrega de vídeo digital de alta qualidade em tempo real, na capacidade de ser escalável para dispositivos modernos, na

possibilidade de ser compatível com várias larguras de banda e, por fim, na possibilidade de ser otimizado para uma maior variedade de *hardware*, tornando-se mais flexível tanto para a vertente comercial como para a particular devido ao facto de deixar de se pagarem taxas de utilização. Em 2014 a Google anunciou que iria proceder ao desenvolvimento do VP10, sendo que, no ano seguinte, começou a libertar o código da primeira versão do VP10 para o público (Larabel, 2018). Este código foi um projeto realizado em conjunto com outro *codec* em desenvolvimento pela Xiph.org/Mozilla intitulado por Projeto Daala e o Thor, criado pela empresa de redes e comunicações Cisco, que tinha como objetivo obter melhores desempenhos nos seus produtos de videoconferência. Apesar dos três projetos terem bastante potencial, falta ainda chegar a um consenso para unir as forças de todos para facilitar e acelerar o desenvolvimento e adoção do novo *codec* a pensar no futuro. Em setembro de 2015, a Google reuniu 6 empresas de grande escala no setor da Internet para, em conjunto, formar uma aliança designada por *Alliance for Open Media* (AOMedia), conforme já referido anteriormente, com o objetivo de desenvolverem *codecs* e tecnologias de nova geração para o interesse público. Os membros fundadores desta aliança incluem as empresas Amazon, Cisco, Google, Intel, Microsoft, Mozilla e Netflix. Em comunicado (AOM, 2015), a aliança referiu que a elaboração de um novo *codec* iria combater algumas das debilidades do projeto WebM, tais como o facto da inovação nos *codecs* estar a ser demasiada lenta e ser ainda bastante fechada. Em meados de 2016 foi o momento em que a aliança lançou a primeira versão beta do projeto, intitulado precisamente por AV1. Juntamente com este anúncio, a AOMedia anunciou que as empresas especializadas em semicondutores, incluindo a AMD, a ARM e a Nvidia, iriam juntar-se à parceria (AOM, 2016).

Com a entrada destas empresas, o novo *codec* começa a ter destinos traçados em questões aplicacionais, desde a implementação em *hardware*, navegadores *Web* e serviços de *streaming online*. No mês de março deste ano, o grupo anunciou a primeira versão estável do *codec* juntamente com o documento característico das especificações dos algoritmos. Os primeiros testes e comparações começaram a surgir nas semanas seguintes, indicando que o AV1 seria capaz de obter melhores resultados na codificação de uma *stream* de vídeo em relação ao antecessor VP9 e competidor direto HEVC (Dataset, Feldmann, Timmerer, & Klagenfurt, 2018).

Apesar dos resultados promissores que este *codec* tem proporcionado, o caminho para a respetiva otimização do mesmo ainda é longo, sendo que o tempo, em média, que o codificador consome para finalizar a compressão de vídeo é muito dilatado, sendo que ainda não pode, pelos menos ainda, ser plenamente utilizado na prática. Investigadores da Mozilla anunciaram que até ao final deste ano, o navegador Mozilla Firefox começaria a integrar por omissão o suporte para AV1 em vídeos que estivessem codificados nesse formato. Devido à sua dimensão e notoriedade, o YouTube criou uma *playlist* disponível para o público de vídeos codificados recorrendo a este *codec* para permitir testar o desempenho do descodificador e, ao mesmo tempo, demonstrar interesse no respetivo projeto de desenvolvimento (YouTube, 2018). A reprodução de vídeos codificados com este *codec* apenas é possível com um navegador *Web* que suporte diretamente o *codec* AV1, como é o caso das versões de desenvolvimento do Mozilla Firefox e do Google Chrome.

O AV1 possui diversas técnicas eficientes para a codificação e descodificação de um vídeo digital. Muitas destas melhorias partem da base deixada pelo seu antecessor VP9, sendo que, ao longo desta análise das especificações técnicas dos *codecs* irá ser efetuada uma comparação a nível qualitativo e quantitativo entre os dois *codecs* em questão.

2.7.1. *Block Partitioning*

Como se pode verificar nas especificações do VP9, este *codec* possui *superblocks* de uma dimensão de *64×64* pixéis, subdividindo-se em 4 formas distintas (2 retângulos vertical/horizontal, quadrado inteiro e divido em 4 blocos quadrados) e de forma recursiva nos quadrados até ao tamanho mínimo de 4×4. O AV1, além de introduzir novas formas de divisão dos blocos, ainda apresenta um tamanho novo para o *superblock*, sendo que agora possui uma dimensão máxima de 128×128 pixéis. Além das 4 formas de divisão que apresentava o seu antecessor, este ainda permite subdividir os elementos em blocos com 2 quadrados e 1 retângulo organizados nas 4 combinações possíveis, e ainda 4 retângulos na vertical e na horizontal. Incluindo-se todas estas inovações, o AV1 possui uma *partition-tree* com 10 estruturas possíveis, como se ilustra na Figura 2.20, podendo subdividir-se de forma recursiva em quadrados até à dimensão mínima de 2×2 em alguns casos. Desta forma permite-se efetuar uma codificação de movimento e textura mais eficiente em regiões de um vídeo em UHD.

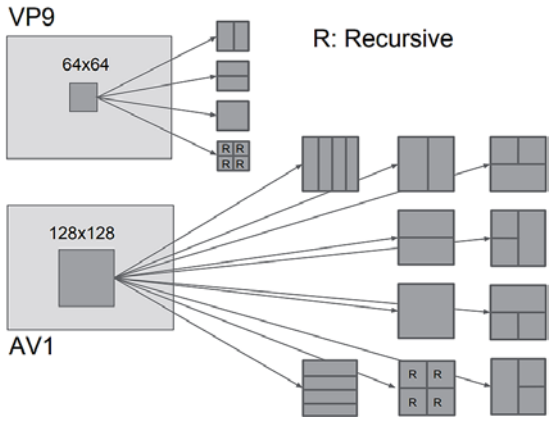

Figura 2.20 – Comparação entre VP9 e AV1 na subdivisão dos blocos (Chen et al., 2018)

2.7.2. *Intra Mode*

O AV1 explora de forma mais profunda as técnicas utilizadas na previsão dos blocos no contexto da codificação espacial. Além dos modos direcionais de previsão, como acontecia no VP9, este *codec* introduz outros tipos de previsão que exploram os blocos com mais detalhe. Além dos 8 ângulos definidos no VP9, este *codec* introduz duas novas variáveis no cálculo, *angle_delta* e *angle_step*, que permitem conjugar melhor com os 8 ângulos nominais para uma maior precisão, permitindo que o total de ângulos possíveis para a previsão seja de 56, fazendo com seja o *codec* com mais modos direcionais espaciais que existe na atualidade.

Além destes, o AV1 introduz 3 novos modos não direcionais: *SMOOTH_V*, *SMOOTH_H* e *SMOOTH* que expandem os blocos na vertical e na horizontal, ou calcula os valores de cada bloco consoante o valor dos blocos adjacentes e de um potencial vizinho que já foi calculado, recorrendo para o efeito a interpolações quadráticas. O modo TM, introduzido no VP9, é substituído neste *codec* pelo preditor *PAETH*, que subtrai o valor do bloco do canto superior esquerdo (que serve de referência para a direção de gradiente) à soma do valor do bloco superior e lateral adjacente. Na Figura 2.21 mostra-se um exemplo simples desta operação. Estes modos servem sobretudo para zonas da *frame* onde ocorrem gradientes, tais como fotografias do céu ou outros tipos semelhantes de fundos de imagens.

Um último modo interessante, que merece ser referenciado, é o *Intra Block Copy*, que permite que o codificador espacial reconstrua um bloco fazendo referência a outro bloco previamente codificado, de forma similar ao que sucede no *Inter Prediction* quando se

faz referência a outras *frames*. Isto é muito benéfico para conteúdos de vídeo que possuem texturas repetidas na mesma *frame*.

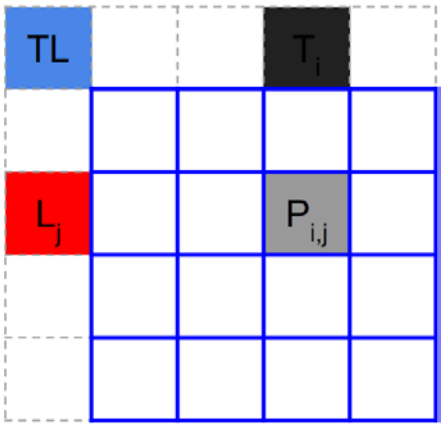

Figura 2.21 – *AV1 Intra Prediction* (Massimino, 2017)

2.7.3. *Inter Mode*

Tal como em outros *codecs* analisados neste trabalho, o cálculo de previsão de *frames* a nível espacial também é efetuado no AV1, mas de uma forma mais eficiente e complexa em relação ao VP9. Um dos motivos pelos quais o AV1 consegue ser mais eficiente do que o seu antecessor é o facto de conseguir utilizar mais *frames* de referência: sete ao invés de apenas três. Em adição às *frames* temporais *LAST* (passado recente), *GOLDEN* (passado distante) e *ALTREF* (futuro) utilizadas pelo VP9, o AV1 adiciona mais duas do passado (*LAST2* e *LAST3*) e duas do futuro (*BWDREF* e *ALTREF2*) como se mostra na Figura 2.22.

Tal como o VP9, este *codec* utiliza um vetor de movimentos fundamental entre *frames* temporais. O vetor *Dynamic Reference Motion Vector Prediction* (REFMV), além de guardar informações obtidas a partir de *frames* vizinhas, também é responsável por proporcionar um mecanismo de previsão para gerar *frames* candidatas temporais. Ainda sobre vetores de movimento, o AV1 implementa, utilizando vetores de movimentos dedicados, uma tecnologia intitulada de *Overlapped Block Motion Compensation* que reduz descontinuidades que possam ocorrer entre os blocos provocadas por erros de compensação.

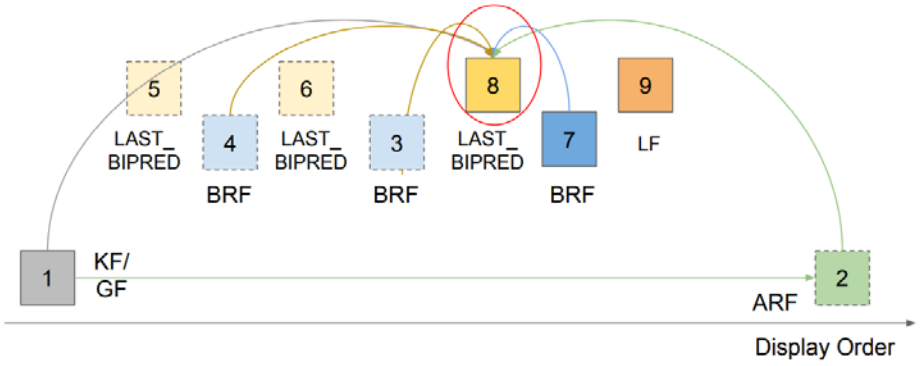

Figura 2.22 – *Frames* de referência utilizadas pelo AV1

2.7.4. *Transform*

No contexto das transformadas, o AV1 inclui ligeiras modificações em relação ao seu antecessor, mas sempre com o objetivo de tornar o processo de compressão e descompressão mais eficiente. O AV1 utiliza vários núcleos de transformadas tanto em blocos do tipo *intra* como *inter*. Consiste num conjunto de 16 combinações verticais e horizontais dos algoritmos das transformadas *DCT*, *ADST* (utilizados anteriormente pelo VP9), *ADST* de forma inversa (*flipADST*) e *IDTX*.

Além disso, o AV1 utiliza blocos de codificação de intensidades de cor (*luma*) em várias dimensões ao invés de uma dimensão fixa, como é utilizada no VP9. É importante destacar que as dimensões retangulares foram igualmente adicionadas neste *codec*, incluindo as seguintes: 4×8, 8×4, 8×16, 16×8, 16×32 e 32×16)

2.7.5. Codificação de entropia

O AV1 implementa a técnica de *Multi-symbol* na codificação de entropia que permite a codificação de símbolos não binários para acelerar os processos de codificação e descodificação. Símbolos hexadecimais permitem uma execução mais eficiente por parte do codificador devido a principalmente utilizarem menos símbolos face aos binários.

No VP9, o processo de quantificação dos coeficientes das transformadas é efetuado de forma sequencial seguindo a ordem da leitura. O modelo probabilístico usado para a obtenção de cada coeficiente baseia-se nos coeficientes anteriormente calculados. O AV1 altera a forma como se efetua esse processo, introduzindo um *Level Map* que captura a distribuição de coeficientes em dois planos níveis de planos de coeficientes. (Chen et al., 2018)

3. Análise e Especificação de um Sistema de *Transcoding*

O sistema proposto neste trabalho tem como base o processo de *transcoding* em tempo real de uma transmissão televisiva de um canal através do protocolo IPTV para dispositivos de tipos diferentes. Os principais objetivos do sistema incluem os seguintes:

- Reduzir a largura de banda necessária por parte dos dispositivos recetores utilizando *codecs* da nova geração e, ao mesmo tempo, permitir a visualização de conteúdos com uma melhor qualidade de imagem em relação a *codecs* anteriores com o mesmo débito binário.

- Centralizar o processo de *transcoding* apenas do lado da operadora, sem que seja necessário que cada estação televisiva adquira novos equipamentos para essa finalidade.

- Adaptar o *codec* de vídeo utilizado consoante a capacidade de descodificação do dispositivo recetor.

Este trabalho descreve uma simulação de uma transmissão televisiva, sendo que não serão utilizadas câmaras para a captura de imagens e, de seguida, a respetiva transmissão. Em vez disso, utiliza-se um conjunto de vídeos previamente gravados com características idênticas àquelas que geralmente são utilizadas numa transmissão televisiva, tais como a resolução da imagem e o *frame rate* do vídeo.

Neste capítulo são estabelecidos os requisitos funcionais, não-funcionais e de sistema adotados para esta implementação. De seguida, descreve-se a arquitetura do sistema que define os componentes necessários para o sistema funcionar em pleno. É feita a definição dos módulos utilizados e especifica-se as respetivas características fundamentais. Numa fase final é feita a definição de alguns cenários que servem como exemplos que podem ser adotados pelo sistema aqui proposto. Esses cenários serão posteriormente testados e avaliados nos capítulos subsequentes.

3.1. Requisitos

3.1.1. Requisitos funcionais

- **RF001:** A reprodução da *stream* de vídeo final deve poder ser efetuada por uma aplicação que suporte os *codecs* em questão.

- **RF002:** A *stream* de vídeo deve possibilitar ser acedida através de um endereço na Web. Ex: rtmp://localhost/live/stream.

- **RF003:** Nas fases de análise de vídeo, através de ferramentas adequadas, deve ser possível analisar todos os dados da transmissão da *stream* a fim de ser possível verificar a respetiva qualidade bem como possíveis perdas.

3.1.2. Requisitos não funcionais

- **RNF001:** A ligação entre o Servidor de *streaming* e o *Transcoder* deve ser cablada para evitar interferências.

- **RNF002:** O *Transcoder* deve dispor dos *codecs* em questão suportados em *hardware* para proceder à codificação e à descodificação de uma forma o mais eficiente possível, reduzindo, ou mesmo evitando, quaisquer atrasos na transmissão.

- **RNF003:** A perda de *frames* de vídeo digital deve ser evitada. A qualidade de vídeo neste projeto é um critério imprescindível e a perda de informação não deve ocorrer.

- **RNF004:** Diminuir ao máximo possível o tempo de atraso entre a *stream* original e a *stream* final.

- **RNF005:** A transmissão do vídeo deve ser efetuada de modo a que os dispositivos possam reproduzir o conteúdo de vídeo digital em várias circunstâncias, incluindo uma ligação cablada, ou via Wi-Fi, ou, ainda, através de uma ligação de dados sobre uma rede celular.

- **RNF006:** O T*ranscoder* deve consultar as capacidades da potência de processamento do recetor para que seja possível escolher o melhor *codec*, de modo a, posteriormente, transmitir a *stream* desejada.

- **RNF007:** O sistema deve estar projetado de modo a ser escalável. Neste caso, a escalabilidade deverá ser proporcional ao número de *streams* de vídeo que é possível transmitir em simultâneo.

3.1.3. Requisitos de sistema

- **RS001:** É necessário um computador para efetuar o processo de *transcoding* com suporte em *hardware* dos *codecs* referidos, nomeadamente o H.264/AVC, o H.265/HEVC e o VP9.

- **RS002:** As operações de *transcoding* são efetuadas através do *software* FFmpeg juntamente com as bibliotecas de *software* associadas.

3.2. Arquitetura do Sistema

A arquitetura deste sistema inclui três módulos distintos, como se ilustra na Figura 3.1, sendo que o módulo de *Transcoder* do vídeo é aquele que tem mais relevância em relação aos restantes já que implementa o objetivo principal do trabalho descrito nesta dissertação. Esta arquitetura representa o fluxo de dados desde a origem do vídeo, passando pela transformação do sinal de vídeo através do processo de *transcoding*, até ao resultado final obtido por diferentes formas. Na Figura 3.2 mostra-se uma representação gráfica da arquitetura do sistema do ponto de vista de uma implementação prática, indicando, de forma clara, quais os intervenientes, substituindo o Servidor de *streaming* pela estação televisiva.

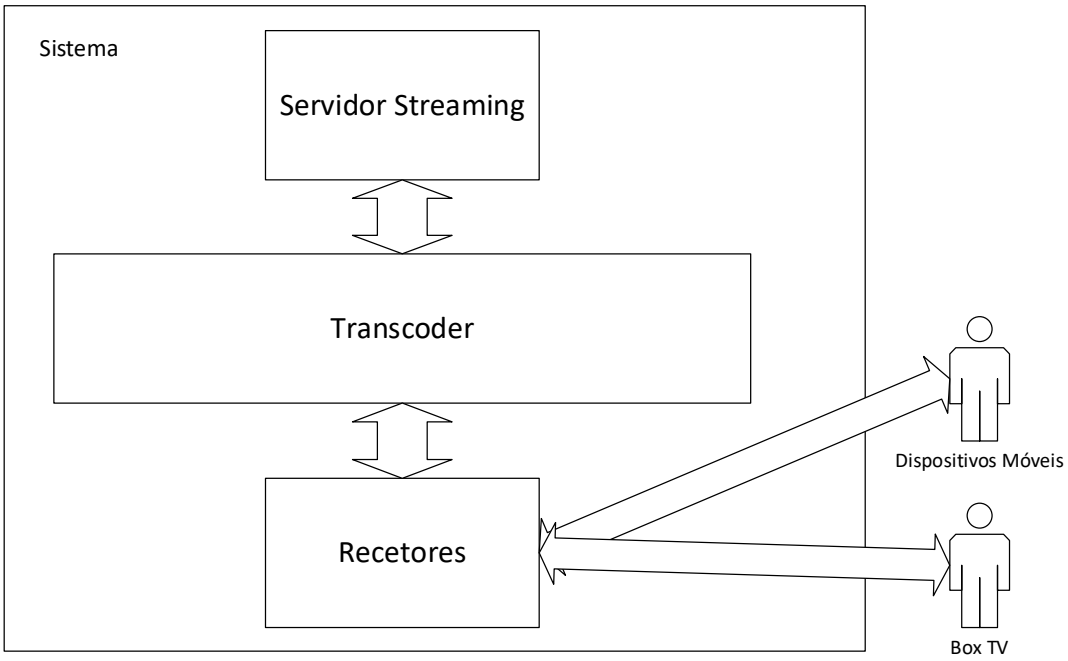

Figura 3.1 – Visão geral da arquitetura do sistema proposto

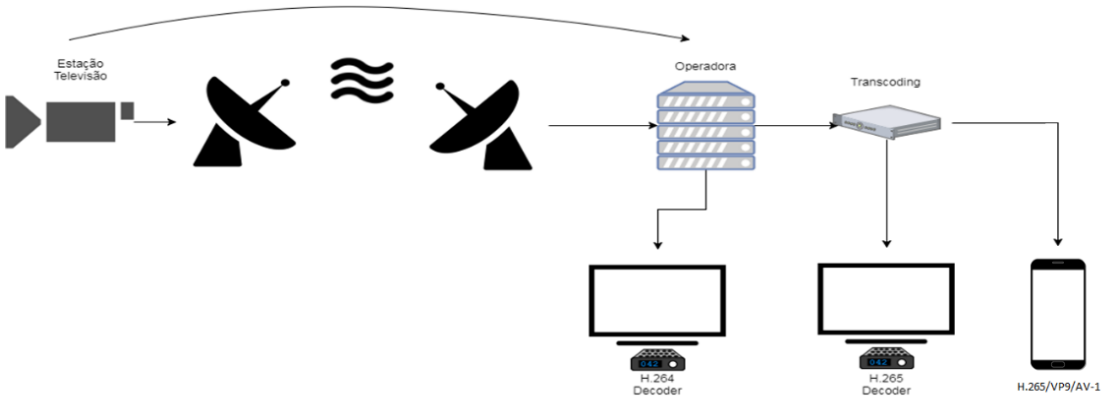

Figura 3.2 – Perspetiva geral de forma gráfica da arquitetura do sistema

3.2.1. Servidor de *streaming* de vídeo

O módulo do Servidor de *streaming* possui uma complexidade muito baixa, visto que tem como principal objetivo o envio de um sinal de vídeo para o *Transcoder*. Este servidor não necessita de dispor de muito poder computacional, apenas deve ter suporte em *hardware* para o *codec* de vídeo H.264 e uma ligação por cabo dedicada para evitar a perda de pacotes durante a transmissão. Na Figura 3.3 pode-se observar as principais funcionalidades deste servidor.

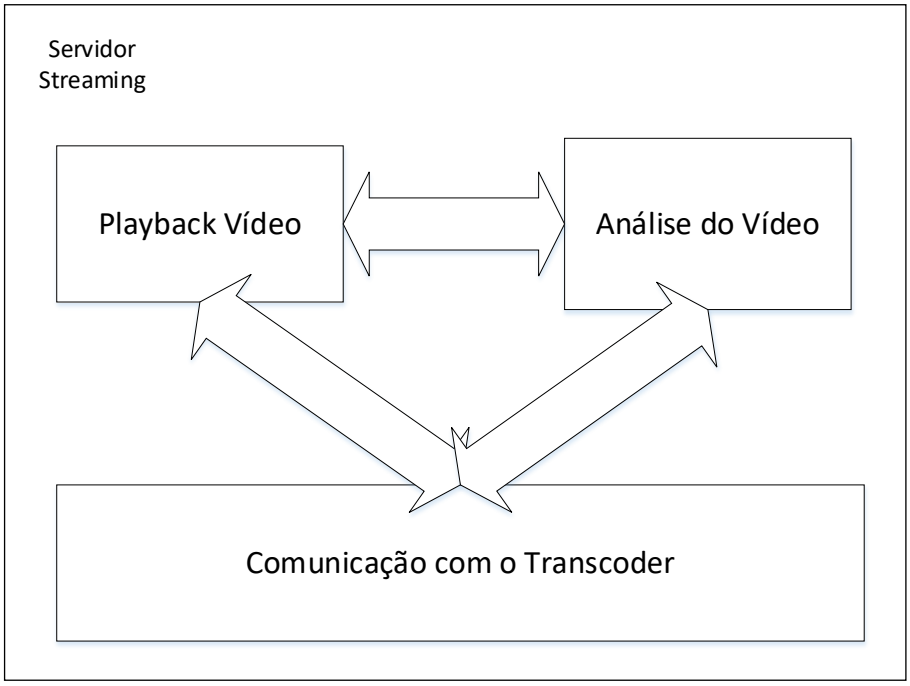

Figura 3.3 – Vista detalhada do servidor *streaming* de vídeo

Mais concretamente, os componentes que integram o módulo do Servidor de *streaming* de vídeo incluem os seguintes:

- ***Playback* do vídeo:** este componente tem como principal funcionalidade o envio de um vídeo codificado em H.264 para o servidor responsável pelo processo de *transcoding* do sinal recebido. É necessário que o sistema envie o vídeo com parâmetros definidos de forma a simular a difusão de uma câmara de vídeo.

- **Análise do vídeo:** este componente realiza uma análise quantitativa e qualitativa da *stream* de vídeo enviada para efetuar uma comparação da qualidade de imagem entre as *streams*, bem como as diferenças das larguras de banda.

- **Comunicação com o exterior:** o servidor deve estar ligado de forma dedicada (preferencialmente por cabo), para ser capaz de enviar a *stream* de vídeo sem perdas e com uma largura de banda suficiente para suportar altas resoluções.

3.2.2. Arquitetura do módulo de *Transcoder*

O módulo de *Transcoder* é responsável pelo processo de transcodificação de uma *stream* de vídeo codificada em H.264 que é recebida pelo Servidor de *streaming* de vídeo para um *codec* de vídeo mais eficiente de nova geração (HEVC, VP9 ou AV1). Para tal, necessita de possuir, preferencialmente, suporte em *hardware* de, pelo menos,

um desses *codecs*, para que a operação de *transcoding* possa ser realizada da forma mais rápida e eficiente possível. Este módulo é igualmente responsável por verificar qual o *codec* suportado pelo recetor que requisitou a *stream* de vídeo. A Figura 3.4 demonstra a arquitetura do módulo de *Transcoder*.

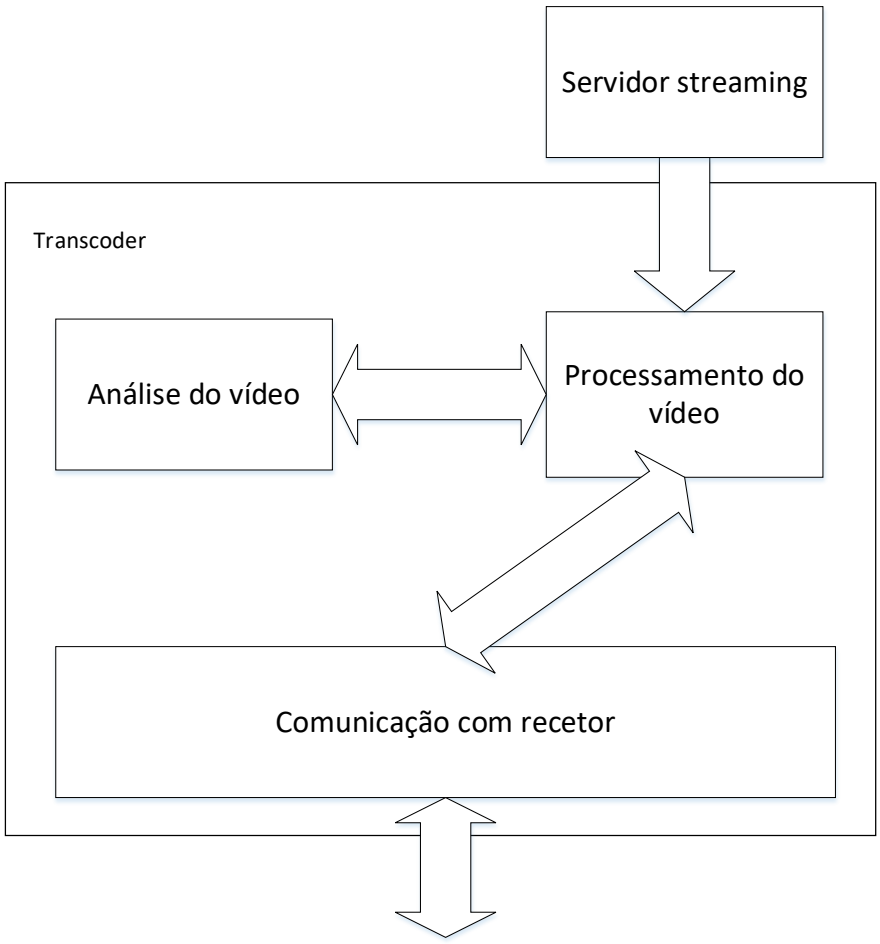

Figura 3.4 – Vista geral da arquitetura do módulo do *Transcoder*

Mais especificamente, os componentes que constituem o módulo do *Transcoder*:

- **Processamento do vídeo:** este componente foi concebido para receber uma *stream* de vídeo codificada em H.264 e efetuar o *transcoding* do vídeo para um *codec* específico consoante o suporte disponível por parte do recetor.

- **Comunicação com o recetor:** esta componente é capaz de comunicar com o recetor final da *stream* de vídeo para definir qual o *codec* a ser utilizado no processo de *transcoding* da *stream* de vídeo. Os resultados são guardados numa base de dados para não ser necessário efetuar a repetição do pedido ao recetor uma segunda vez.

Análise do vídeo: tal como no Servidor de *streaming* de vídeo, este componente efetua uma análise qualitativa e quantitativa da *stream* de vídeo final para avaliar possíveis erros e falhas, como também as diferenças substanciais entre a *stream* de vídeo original e a final.

3.2.3. Recetores

Este sistema contém dois tipos distintos de utilizadores finais, nomeadamente as *TV boxes* e os dispositivos móveis que apenas têm como objetivo principal receber a *stream* de vídeo mais adequada ao respetivo equipamento. A Figura 3.5 indica, de forma simplificada, a interação entre o servidor responsável pelo *transcoding* da *stream* de vídeo e um determinado dispositivo recetor.

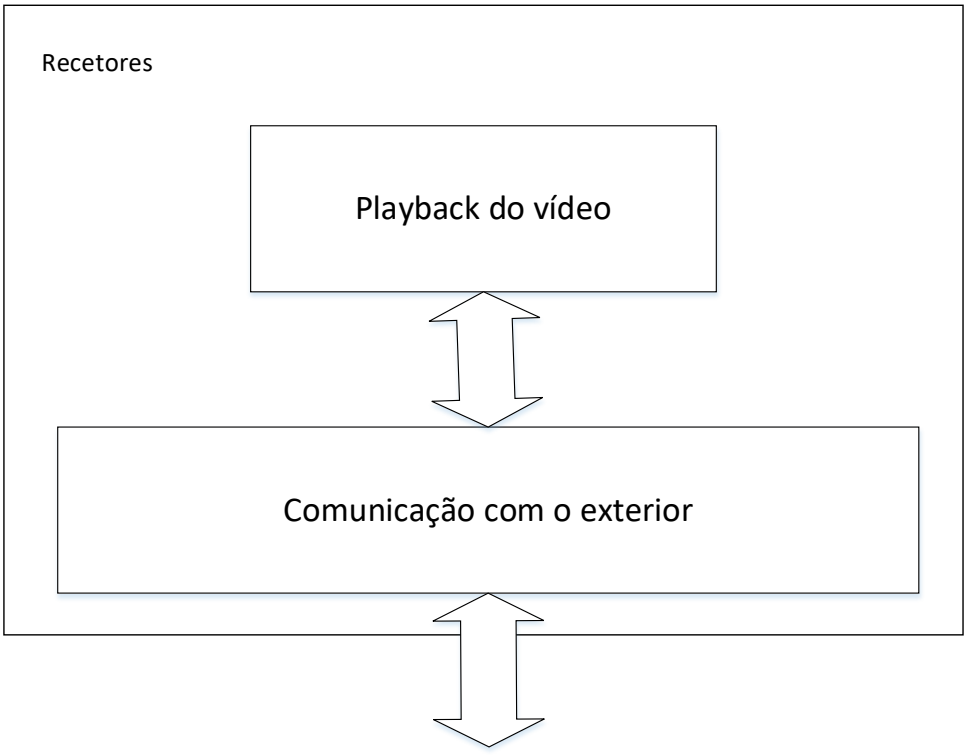

Figura 3.5 – Vista geral das funcionalidades do recetor

Os componentes que constituem os módulos recetores do Sistema de *transcoding* incluem os seguintes:

- **Comunicação com o *Transcoder*:** este componente tem como objetivo comunicar com o Servidor de *transcoding* com duas finalidades distintas: enviar, na sua primeira utilização, informação relativas ao suporte, a nível de *codecs*, do dispositivo, e receber a *stream* de vídeo adaptada para o dispositivo em questão.

- **Playback do vídeo:** os dispositivos recetores devem de possuir uma aplicação que seja capaz de receber e reproduzir a *stream* de vídeo enviada pelo *Transcoder*.

3.3. Contexto do utilizador

Neste projeto existem dois tipos de utilizadores a tomar em consideração: o primeiro tipo de utilizador possui uma *TV box* com ligação à Internet que é capaz de receber conteúdo televisivo através do protocolo IPTV, e o outro é um tipo de utilizador que deseja visualizar canais televisivos através de um dispositivo móvel tal como um *smartphone* ou um *tablet*. O utilizador que possui a *TV box* possui uma conexão à Internet com uma largura de banda limitada, com baixos valores de débitos, o que impossibilita a visualização de canais em alta definição codificados com o *codec* H.264. O utilizador que recorre a um dispositivo móvel, geralmente possui uma ligação móvel à Internet que, geralmente, resulta em valores instáveis da largura de banda podendo levar a paragens constantes ou à redução da resolução do vídeo durante a reprodução de um canal televisivo, precisamente para evitar tais paragens constantes. Em ambas as partes, a qualidade do vídeo é afetada, prejudicando a visualização dos utilizadores desse tipo de conteúdos.

3.4. Cenários de utilização do sistema proposto

- **Processo de *transcoding* para uma *TV box* com suporte H.265**
    - O utilizador recebe uma *stream* codificada com o *codec* H.265 numa *TV box* certificada e licenciada para o suporte para esse *codec* com ligação à Internet.

- **Processo de *transcoding* para um dispositivo móvel (Android ou iOS)**
    - O utilizador possuí um dispositivo móvel com capacidades de reproduzir vídeo em alta definição. Dependendo do sistema operativo em questão e do modelo do *smartphone*, o dispositivo deverá possuir suporte para o *codec* VP9, HEVC ou até ambos em alguns casos. Neste caso o sistema avalia o desempenho e possíveis erros que possam ocorrer num ambiente móvel.

4. Implementação do protótipo do Sistema de *Transcoding*

4.1. Introdução

Este capítulo especifica com detalhe a forma como foram implementados, isto é, montados e configurados os módulos do Sistema de *Transcoding* especificado anteriormente, incluindo o módulo do Servidor de *streaming* de vídeo, o módulo do *Transcoder* e os recetores finais, bem como os respetivos componentes.

Na primeira secção, que aborda o módulo do Servidor de *streaming* de vídeo, efetua-se uma análise a possíveis conteúdos que podem ser transmitidos, bem como a um conjunto de ferramentas necessárias, em termos de *software* e *hardware*, para enviar uma *stream* de vídeo para o módulo de T*ranscoder*.

A secção seguinte, sobre o servidor responsável pelo *transcoding* das *streams* de vídeo recebidas pelo Servidor de *streaming* de vídeo, detalha as ferramentas necessárias para receber a *stream* de vídeo e para efetuar a respetiva transcodificação, assim como os parâmetros necessários para se aplicarem algumas otimizações. Para além dessas ferramentas, efetua-se igualmente uma exposição teórica sobre as bibliotecas dos *codecs* implementadas em *software* e em *hardware* e também se identificam os protocolos de rede utilizados para a transmissão das *streams* de vídeo

Mais adiante, a secção dos recetores indica detalhadamente quais as plataformas que possibilitam a reprodução dos *codecs* de vídeo estudados, assim como o ambiente tecnológico em que os dispositivos se encontram.

Finalmente, a última secção dedica-se às ferramentas utilizadas para quantificar as diferenças entre as *streams* de vídeo, indicando os respetivos propósitos para este trabalho.

4.2. Módulo do Servidor de *streaming* de vídeo

O módulo do Servidor de *streaming* de vídeo é responsável por transmitir uma *stream* de vídeo com características bem definidas para o *Transcoder*, de forma a simular o

processo de gravação e o envio das imagens adquiridas num estúdio televisivo. No contexto deste trabalho, o servidor transmite de forma contínua um conjunto de conteúdos de vídeo de formatos diferentes para permitir avaliar, posteriormente, os resultados obtidos após o processo de *transcoding* da *stream* de vídeo.

4.2.1. Repositório de vídeos

Na grelha de canais televisivos disponibilizada pelos fornecedores de televisão existem canais com conteúdos de vídeo com aspetos diferentes dependendo da finalidade da estação televisiva específica. Por um lado, canais de conteúdos informativos tendem a proporcionar imagens com fundos estáticos, o que acelera o processo de codificação devido à baixa complexidade de tais imagens. Por outro lado, existem canais de caráter desportivo ou cinematográfico que contêm imenso movimento e imprevisibilidade, aumentando a complexidade do codificador e, geralmente, aumentando igualmente o débito binário para estes tipos de vídeo.

Para simular essa diversidade de conteúdos transmitidos na televisão foi necessário recorrer à pesquisa de bases de dados de conteúdos multimédia de forma a descarregar os vídeos para o servidor. O repositório principal de vídeos neste projeto é o da fundação Xiph.Org[11]. Esta fundação dedica-se inteiramente a proteger, dos setores privados, as fundações dos conteúdos multimédia na Internet. Este grupo é o responsável pelo desenvolvimento de *codecs* de áudio como o OPUS, conhecido pela sua utilização em plataformas tais como o YouTube, devido a ser gratuito e livre de taxas de utilização (para combinar com o *codec* de vídeo VP9), e também o FLAC, utilizado essencialmente para reter o máximo da qualidade do áudio devido ao facto de ser um *codec lossless* de elevada eficiência (sem perdas). Para além do áudio, a fundação Xiph desenvolveu *codecs* de vídeo sendo que um em particular (Daala) serviu de base para o desenvolvimento do *codec* AV1, tal como foi como abordado na secção 2.7. O repositório da fundação, como se pode observar na Figura 4.1**,** contém sequências de vídeos fornecidas por diversas entidades de forma livre, sendo que qualquer indivíduo pode proceder ao descarregamento do conteúdo sem quaisquer custos. Os conteúdos armazenados no repositório estão codificados sem quaisquer perdas de informação no formato YUV4MPEG (.y4m) de forma a realizar a codificação destes conteúdos para o *codec* de vídeo pretendido, como é o caso do H.264 no caso

[11] https://www.xiph.org/

deste trabalho. O repositório encontra-se dividido em três secções: a primeira abrangendo conteúdos SD num aspeto de 4:3, a segunda, contendo conteúdos HD e UHD em 16:9 e, por fim, conteúdos UHD com uma profundidade de 10 bits (HDR). Em cada sequência de vídeo incluído neste repositório é apresentado, sempre que possível, o número de *frames* que o vídeo contém, o respetivo *aspect ratio* e a resolução, juntamente com a dimensão do espaço de armazenamento ocupado em disco. Neste trabalho será utilizada uma sequência de cada formato de forma a testar os *codecs* abordados nesta dissertação e simular parte do conteúdo transmitido na televisão. As sequências específicas que foram escolhidas para os testes encontram-se definidas no Capítulo 5, na parte da dissertação que descreve o estudo empírico que foi realizado.

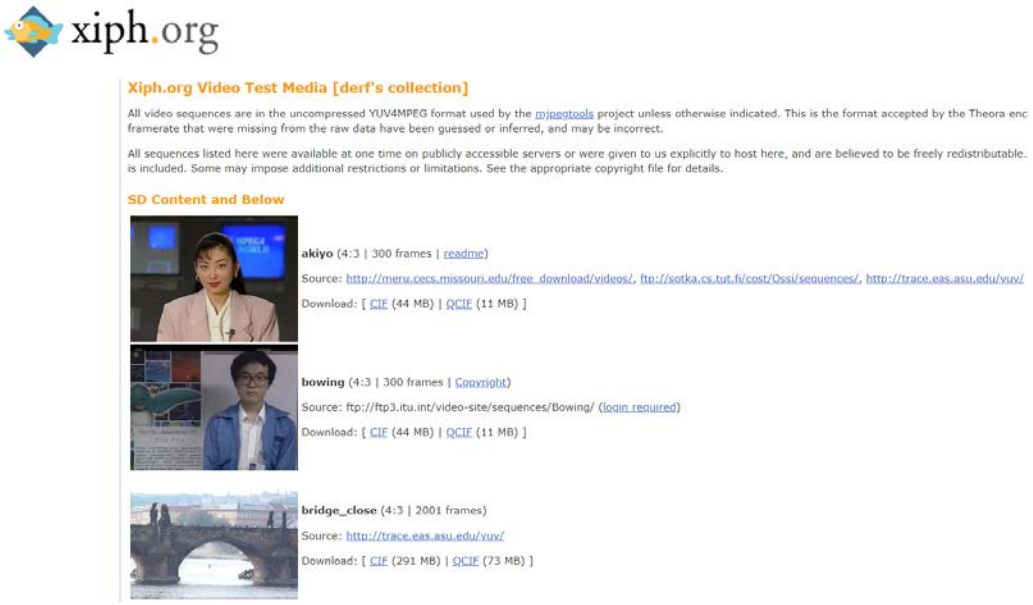

Figura 4.1 – Repositório de vídeos fornecido pela fundação Xiph.org

4.2.2. Configurações e parâmetros

Após descarregar as sequências de vídeo necessárias para a realização dos testes previstos neste trabalho, é necessário codificá-las de forma a que cumpram certos requisitos, semelhantes aos que são utilizados numa transmissão televisiva, de modo a permitir-nos simular, o melhor possível, as condições de uma transmissão de sinal de TV. O vídeo é inicialmente codificado em H.264 e serão utilizados vídeos progressivos com dois *frame rate*: 25 fps e 50 fps, para ir de encontro às normas utilizadas na União Europeia, como foi revisto e justificado detalhadamente na secção 2.2.

O *software* que foi selecionado para ser utilizado no contexto do componente de configurações e parâmetros é o FFmpeg[12], que será utilizado tanto no processo de codificação das sequências de vídeo descarregadas, como no processo de envio da *stream* de vídeo para o *Transcoder*. Os detalhes sobre esta ferramenta de *software* e dos respetivos parâmetros serão expostos a seguir, nomeadamente no que diz respeito ao *Transcoder*, visto que este módulo possui um grande impacto no presente projeto.

As codificações das sequências de vídeo são efetuadas através do FFmpeg recorrendo à linha de comandos, independentemente do Sistema Operativo utilizado para o efeito (MS Windows ou uma distribuição Linux). A Figura 4.2 mostra um exemplo do comando necessário para codificar uma sequência de vídeo sem perdas para a mesma sequência comprimida no formato H.264.

```
ffmpeg -i 720p5994_stockholm_ter.y4m -r 25 -c:v libx264 -crf 0 stockholm_720_25fps.mp4
```

Figura 4.2 – Codificação de uma sequência de *frames* no formato y4m para um vídeo H.264

O comando utilizado neste processo encontra-se dividido em quatro segmentos (ou parâmetros) distintos incluindo os seguintes:

- "-i" – parâmetro que identifica o ficheiro armazenado localmente que irá ser processado pelo FFmpeg.

- "-c:v" – parâmetro que define a biblioteca de *software* que é utilizada para o processo de codificação do vídeo definido na linha anterior. Dado que o dispositivo, neste contexto, possui capacidade de processamento suficiente, e o *delay* não é problemático, a biblioteca escolhida será a *libx264*. O processo será efetuado por *software* - (na realidade, o processo torna-se mais demorado do que se fosse efetuado por *hardware*, mas proporciona resultados mais positivos em termos de eficiência da codificação).

- "-crf" – este parâmetro identifica o valor a utilizar da *Constant Rate Factor* que se trata de uma constante muito comum para os *codecs* de vídeo utilizados neste projeto. O valor atribuído neste parâmetro, no contexto da *libx264*, pode variar entre 0 e 51, sendo que valores mais baixos resultam numa compressão com mais qualidade de imagem e valores mais altos comprimem os vídeos de forma a

[12] https://www.ffmpeg.org/

ocuparem menos espaço de armazenamento, mas reduzindo a qualidade de imagem. Nos testes que foram realizados no âmbito deste trabalho, foi utilizado o valor 0 para se garantir que se perde o mínimo de informação possível do vídeo original durante o processo de codificação para H.264.

- "-r" – este é o parâmetro que define o *frame rate* para o vídeo final codificado. Neste caso o valor é 25 fps ou 50 fps em alguns casos para ir de encontro às normas utilizadas na União Europeia.

- "**nome_do_ficheiro**.mp4" – por fim, neste parâmetro indica-se o nome do ficheiro de saída e o respetivo formato contentor. Existem vários *containers* de vídeo que suportam a codificação H.264. O formato contentor utilizado neste projeto é o MP4.

Ao codificar as sequências de vídeo escolhidas encontraram-se algumas diferenças, conforme se pode observar em alguns exemplos ilustrados na Figura 4.3. As diferenças dizem respeito à dimensão do espaço ocupado em disco pelos vídeos originais sem perdas (formato RAW) e pelas respetivas sequências codificadas em H.264. Após a realização deste processo, os vídeos codificados são enviados através do FFmpeg, seguindo a mesma metodologia utilizada processo anterior.

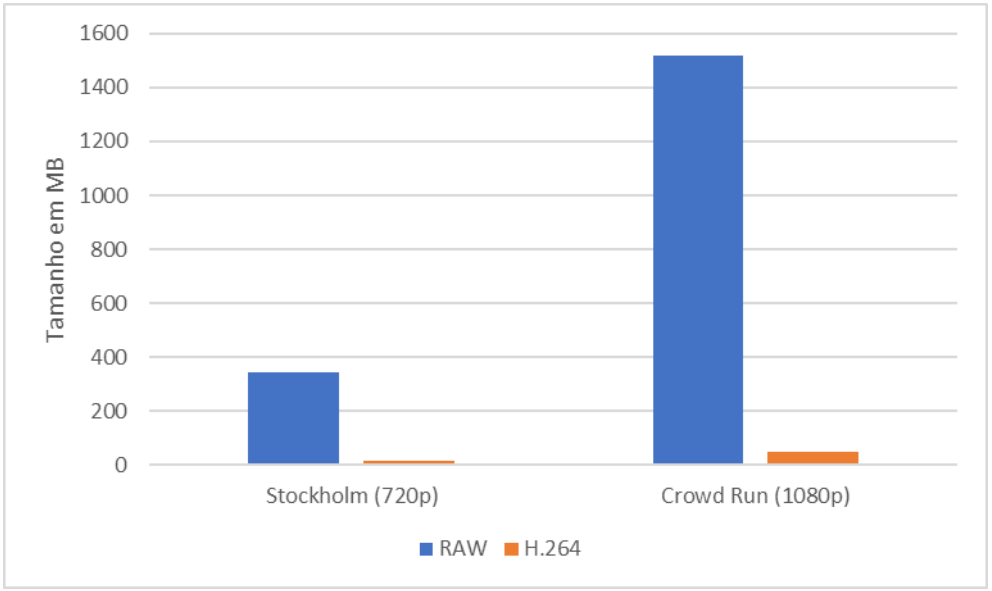

Figura 4.3 – Diferença de tamanhos entre vídeos não comprimidos e comprimidos (H.264)

Assim, prosseguindo com o envio dos vídeos codificados através do FFmpeg, novos parâmetros são adicionados face ao exemplo anterior e outros são modificados tal como se detalha a seguir:

- "-re" – este novo parâmetro indica ao FFmpeg para ler o vídeo à mesma velocidade do *frame rate* do vídeo indicado no *input*. Este parâmetro permite, deste modo, simular um dispositivo de gravação de imagens tal como uma câmara de vídeo profissional.

- "-c copy" – este novo parâmetro faz com que o vídeo introduzido não sofra quaisquer alterações e apenas seja copiado para o destino final, visto já ter sido codificado anteriormente de acordo com os requisitos necessários.

- "-f rtsp rtsp://ip_address" – para além de ser possível armazenar o resultado final num ficheiro local, é igualmente possível transmitir a *stream* resultante de vídeo para um determinado endereço IP através de um determinado protocolo que é aqui especificado. Neste trabalho utiliza-se o protocolo RTSP para o envio da *stream* de vídeo para o módulo de *Transcoder*.

O resultado deste processo encontra-se ilustrado na Figura 4.4.

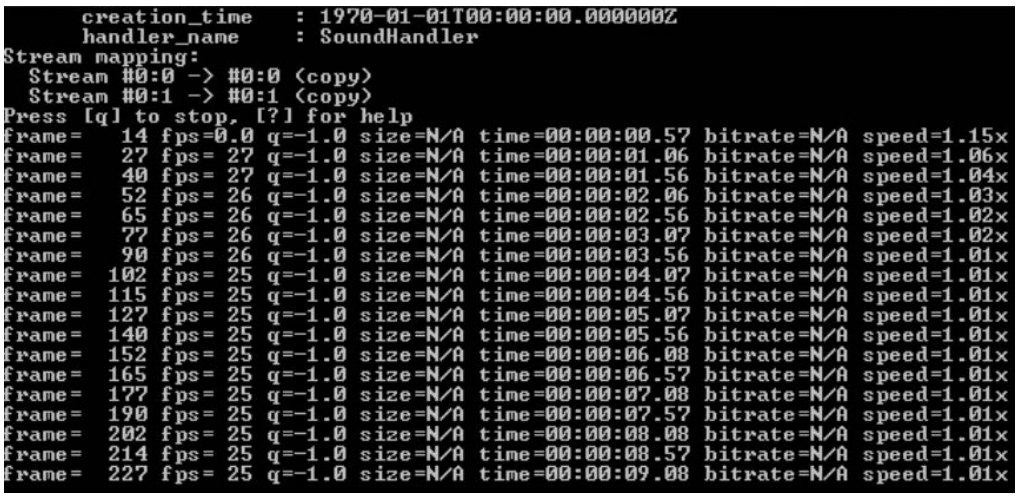

Figura 4.4 – Informação produzida durante a transmissão da *stream* de vídeo

4.2.3. Dispositivo

Para garantir que o módulo do Servidor de *streaming* de vídeo realize a tarefa pretendida de uma forma competente, é necessário que o equipamento obedeça a quatro requisitos fundamentais: (i) dispor de armazenamento suficiente para guardar as sequências de vídeo codificadas, (ii) proporcionar um acesso rápido e eficiente de modo a diminuir ao mínimo possível o tempo de acesso (*delay*), (iii) proporcionar um processamento gráfico com suporte em *hardware* para, pelo menos, o *codec* de vídeo H.264 de modo a ser capaz de codificar de forma eficiente os vídeos descarregados proveniente do repositório Xiph.org e, por fim (iv) proporcionar uma ligação cablada à rede para assegurar uma largura de banda local de cerca de 100 Mbps e evitar possíveis erros de transmissão, tais como aqueles que ocorrem normalmente nas ligações sem fios (por WiFi). Neste trabalho foi utilizado um PC tal como o que ilustra na Figura 4.5.

Figura 4.5 – Computador utilizado para simular o Servidor de *streaming*

4.3. Módulo do *Transcoder*

A função principal de um *Transcoder* passa pela conversão de uma *stream* de vídeo digital com um determinado formato ou débito binário para uma *stream* de vídeo digital codificada num outro formato ou com outro débito binário distinto do original (W. Barz & A. Bassett, 2016). Neste trabalho, o *Transcoder* é utilizado com o objetivo de receber uma *stream* de vídeo codificada previamente em H.264 pelo módulo do Servidor de *streaming* de vídeo e efetuar a conversão dessa sequência H.264 numa outra sequência codificada com outro *codec* de vídeo, seja o HEVC, seja o VP9, dependendo das características do recetor final.

4.3.1. *Software* e bibliotecas

Tal como foi indicado anteriormente, neste projeto, o módulo do *Transcoder* também utiliza a ferramenta FFmpeg como o *software* responsável pela receção da *stream* de vídeo H.264 e pela respetiva transcodificação para o formato pretendido (HEVC ou VP9).

Neste contexto, é importante referir que o FFmpeg designa um projeto responsável pelo desenvolvimento de uma ferramenta de *software* gratuita e *open-source* que tem como objetivo disponibilizar uma *framework* multimédia de bibliotecas de *software* escritas em C que são capazes de efetuar um conjunto de operações essenciais sobre vídeo digital, tais como codificar, descodificar, aplicar filtros e efetuar o *streaming* de conteúdos multimédia. Por este motivo, este *software* foi escolhido para a parte de implementação deste trabalho, devido a englobar todas as bibliotecas de *software* necessárias às operações previstas, e ainda por dispor de suporte em vários sistemas operativos, ao mesmo tempo que é gratuito e proporciona muita informação e documentação sobre os componentes que a integram.

O FFmpeg é constituído por vários componentes dos quais se destaca o *libavformat* que se trata de uma biblioteca de *software* responsável por conter *demuxers* e *muxers*[13] para efetuar as operações de leitura das *streams* de vídeo e áudio codificadas. O *libavcodec* é outro dos componentes principais do FFmpeg, sendo que este contém as bibliotecas de *software* mais populares para a utilização dos *codecs* de áudio e vídeo, como são os casos das *libx264* e *libx265* para, respetivamente, o H.264 e o H.265/HEVC e as bibliotecas *libvpx* e *libaom* para, respetivamente, o VP9 e o AV1. A Figura 4.6 ilustra os passos envolvidos no processo de *transcoding* que se realiza através do FFmpeg.

[13] *Demuxer* é a forma simplificada de *Demultiplexer*. É o *software* responsável por separar os diferentes medias incluídos numa *stream* multimédia para serem posteriormente processados separadamente. *Muxer* designa o processo inverso de multiplexagem dos vários conteúdos de media numa *stream* única que é enviada para o recetor.

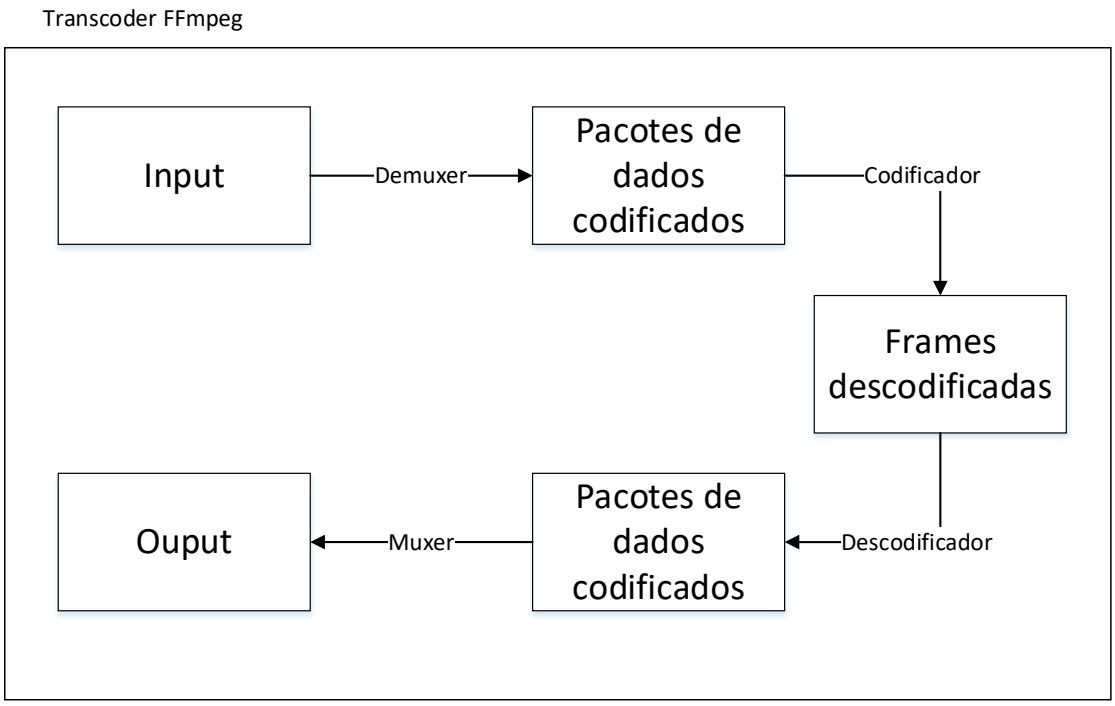

Figura 4.6 – Vista geral do funcionamento do T*ranscoder* na ferramenta FFmpeg

A escolha das bibliotecas de *software* a utilizar no FFmpeg é um tópico essencial no presente trabalho já que o módulo de *Transcoder* é a parte do sistema que desempenha a função principal no processo de transcodificação de uma *stream* de vídeo. De acordo com a forma como são implementadas, existem duas categorias distintas de bibliotecas de *codecs* de vídeo no FFmpeg:

- Implementação em *Software* – Neste caso, o código das bibliotecas do FFmpeg é interpretado exclusivamente através do CPU. De uma forma geral, estas bibliotecas são desenvolvidas constantemente através de comunidades muito ativas que têm como objetivo tornar o codificador mais eficiente ao longo do tempo. Por possuírem um grande suporte, estas bibliotecas possuem melhores resultados a nível de qualidade de imagem quando comparadas com as bibliotecas que são implementadas em *hardware*. No presente trabalho, as bibliotecas implementadas em *software* que são utilizadas no módulo de *Transcoder* incluem a *libx264* para o H.264, a *libx265* para o H.265, a *libvpx* para o VP9 e, finalmente, a *libaom* para o AV1 .

- Implementação em *Hardware* – neste segundo caso, são muitos os fabricantes que oferecem acesso a plataformas com *hardware* dedicado a tarefas relacionadas com o processamento de informação multimédia, mais

especificamente, com o processamento de vídeo digital. Tais fabricantes implementam algumas normas de codificação e descodificação de vídeo digital nas próprias plataformas. Assim, para aceder às funcionalidades existentes no *hardware*, são necessárias API[14] capazes de estabelecer a comunicação entre os comandos do Sistema Operativo e a respetiva plataforma. Este processo é capaz de produzir resultados equivalentes aos que se obtêm com implementações em *software*, mas utilizando menos energia e menos poder de processamento, já que não recorrem tão intensivamente ao CPU. No entanto, a desvantagem desta implementação é o facto de não permitir muita flexibilidade no que diz respeito às configurações dos *codecs*, uma vez que qualquer implementação em *hardware* se encontra restringida às funcionalidades lançadas até àquele período de tempo e não ser impossível atualizar tais funcionalidades, a não ser que se incluam as atualizações num novo produto de *hardware*. Por exemplo, a Intel dispõe da biblioteca *libmfx* que comunica com as diversas plataformas do mesmo fabricante. Os *codecs* de vídeo suportados por esta biblioteca estão dependentes do modelo do *chip* gráfico implementado pelo fabricante. A Nvidia proporciona implementações, de forma muito semelhante, através da biblioteca NVENC e, finalmente, a AMD proporciona implementações através da biblioteca VAAPI.

Um dos requisitos principais deste projeto inclui a possibilidade de utilizar codificadores baseados em *hardware* para acelerar o processo de *transcoding* da *stream* de vídeo, mesmo que o resultado final implique sacrificar um pouco a qualidade de imagem quando comparada com a que se obtém comos codificadores baseados em *software*. Em essência, os codificadores em *hardware* são mais eficientes no que diz respeito ao tempo consumido para a transcodificação e ao baixo consumo energético que evidenciam ao efetuar o processo, dado que utilizam o *chip* gráfico que contém os *codecs* de vídeo embutidos na plataforma.

Para a utilização com base em *hardware*, a API escolhida no âmbito deste trabalho foi a da Intel (*libmfx*), dado que os equipamentos testados contêm *chips* gráficos da Intel com suporte para *codecs* de vídeo, tais como o H.264 e o H.265. O equipamento utilizado

[14] API – provém do acrónimo inglês *Application Programming Interface*. Trata-se de um conjunto de ferramentas de *software* e protocolos que proporcionam o acesso às funcionalidades de um determinado *software* sem que este esteja envolvido nos detalhes de implementação mas apenas nos serviços fornecidos.

neste projeto não possui suporte para o *codec* VP9, sendo que este último será testado num computador com maior capacidade computacional para efetuar o processo por *software* com parâmetros adequados para o propósito de *streaming*. No que diz respeito ao *codec* AV1, por se encontrar ainda numa fase embrionária de desenvolvimento inicial, nenhum fabricante incluiu este *codec* nos seus equipamentos. Assim, neste trabalho, o *codec* AV1 será testado por *software* através da biblioteca *libaom* disponível no FFmpeg.

A *libmfx* é uma biblioteca de *software* proprietária da Intel para utilizar em plataformas da fabricante que sejam compatíveis com os *codecs* implementados. Para proceder à utilização da respetiva API, é necessário estabelecer a escolha do Sistema Operativo. De acordo com a documentação das API e plataformas compatíveis com o FFmpeg (FFmpeg, 2018), a *libmfx* é compatível com as plataformas MS Windows e distribuições Linux, sendo que no MS Windows a biblioteca é facilmente acedida através do FFmpeg desde que os *drivers* de vídeo estejam devidamente instalado, permitindo a utilização de mais funcionalidades face às bibliotecas DXVA2/D3D11VA desenvolvidas pela própria Microsoft. Estas últimas bibliotecas apenas dispõem de descodificadores enquanto que a *libmfx* dispõe de codificadores e descodificadores para além de outras funcionalidades adicionais.

Já a utilização da *libmfx* em plataformas Linux implica a instalação e configuração de vários módulos, incluído o *software* proprietário Intel Media Server Studio, que dispõe dos *drivers* de vídeo adequados, para além de outras ferramentas de análise. Neste trabalho, foi realizada a instalação e a configuração da *libmfx* na distribuição Linux CentOS 7.2.

Todos os procedimentos efetuados no contexto deste trabalho estão de acordo com as indicações descritas num artigo publicado pela própria Intel (Intel Corporation, n.d.) que define os detalhes dos quatros passos essenciais para testar a API de acesso à *libmfx*. Após a instalação e a configuração do Sistema Operativo, procede-se às instalações das ferramentas de *software* necessárias, incluindo o *Intel Media Server Studio*, seguido da última versão estável da ferramenta FFmpeg, neste caso a versão 4.1. A desvantagem associada a esta forma de proceder prende-se com a limitação a nível financeiro associada à utilização deste tipo de plataforma. A Intel dispõe de duas versões do Intel Media Server Studio: a *Community Edition* e a *Professional Edition*,

sendo que a versão para a comunidade é gratuita mas não permite o acesso ao codificador para a norma HEVC. Já a versão profissional disponibiliza uma versão gratuita de testes de 30 dias, proporcionando os codificadores necessários para o projeto, mas com um número limitado de *frames* a codificar, não sendo, por isso, apropriada para o presente trabalho. Em suma, a configuração da API de acesso à *libmfx* da Intel foi efetuada com sucesso, tendo-se instalado todas as bibliotecas necessárias para a implementação do protótipo do sistema definido neste trabalho, tal como se ilustra na Figura 4.7. Apesar de não ter sido utilizada a plataforma da Intel no Linux, fica a demonstração do que seria alcançável em termos teóricos.

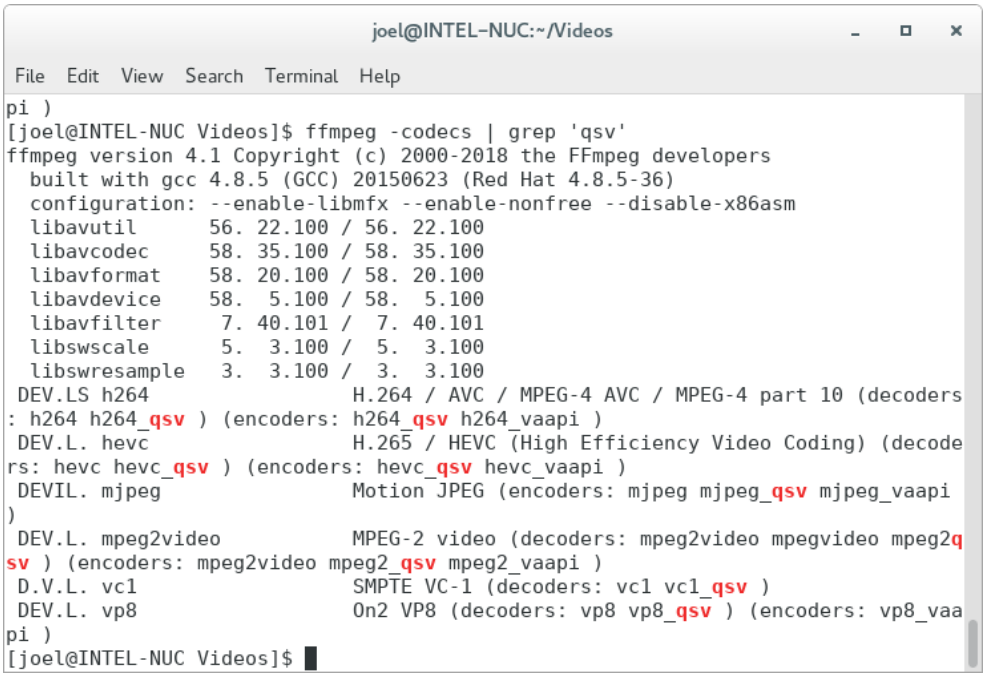

Figura 4.7 – Lista de *codecs* da *libmfxda Intel* incluídos na implementação do protótipo do Sistema proposto

Neste trabalho, a *libmfx* está incluída na versão do FFmpeg utilizada (versão 4.1) e é acessível através das API que se encontram disponibilizadas através dos *drivers* de vídeo da Intel que são instalados juntamente com o respetivo *hardware*.

Além do FFmpeg e das respetivas bibliotecas de *software*, foi igualmente necessário recorrer à utilização de uma ferramenta de *software* adicional que fosse capaz de simular um Servidor de *streaming* de conteúdos multimédia. De facto, após a conclusão do processo de *transcoding* da *stream* de vídeo, a *stream* transcodificada resultante deve ser transmitida para um servidor multimédia de forma a que fique disponível para o acesso dos vários dispositivos recetores, incluindo os dispositivos móveis (*smartphones*

ou *tablets*), os dispositivos fixos (computadores de secretária) ou ainda as *TV boxes*. A escolha da ferramenta adicional de *software* a utilizar para este projeto recaiu no motor *Wowza Streaming Engine* desenvolvido pela *Wowza Media Systems*. Esta ferramenta de *software* foi escolhida por vários motivos, sendo que os dois principais passam pelo suporte diretamente no *software* dos protocolos de rede e de todos os *codecs* utilizados neste projeto. O *Wowza* proporciona também configurações para o permitir atuar como *Transcoder*, mas entendeu-se não ser a escolha mais indicada devido ao facto deste motor ser muito limitado em termos das bibliotecas de *codecs* disponíveis e respetivos parâmetros de configuração, tendo-se dado preferência à ferramenta FFmpeg, conforme justificado anteriormente. Assim, no contexto deste trabalho, este motor de *software* é utilizado apenas para receber a *stream* de vídeo gerada pelo *Transcoder* através do FFmpeg, e funcionar como Servidor de *streaming*, estando disponível em rede num determinado endereço para ser acedido pelos dispositivos recetores. Por fim, outra funcionalidade útil para este projeto que é facultada pelo *Wowza* é a possibilidade de monitorizar a rede indicando de forma gráfica, ou estatística, a largura de banda consumida pelas várias *streams* de vídeo, à entrada e à saída, indicando o respetivo protocolo aplicacional utilizado pela *stream* de vídeo (Figura 4.8).

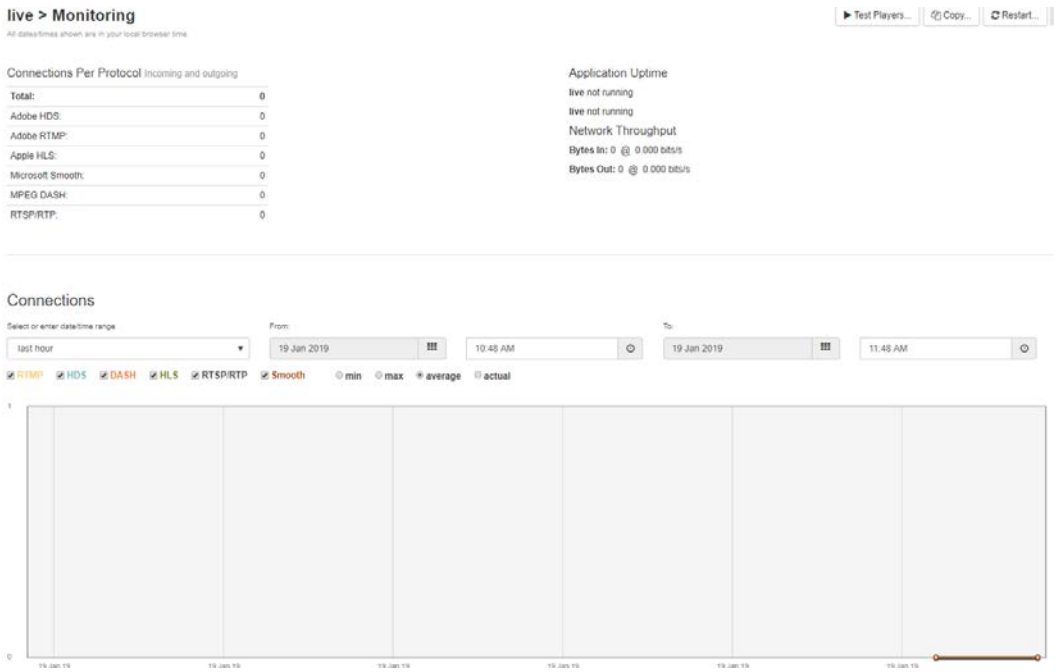

Figura 4.8 – Monitorização da rede no Wowza

4.3.2. Parametrização da ferramenta FFmpeg

A parametrização nas configurações do FFmpeg é essencial para o normal funcionamento do processo de *transcoding* no Sistema proposto neste trabalho. Consoante a biblioteca de *software* utilizada, é necessário aplicar diferentes parâmetros, devido a possuírem configurações diferentes. O modo como a configuração do FFmpeg é efetuada no *Transcoder* é muito semelhante ao procedimento utilizado previamente no caso da configuração do módulo de Servidor de *streaming*. Alguns parâmetros adicionais incluídos nesta configuração servem para controlar a largura de banda da *stream* de vídeo e a respetiva qualidade de vídeo. A sintaxe do comando base executado no T*ranscoder* respeita a seguinte formatação:

FFmpeg –i <endereço ip do servidor streaming> <parâmetros do FFmpeg> -c:v <codec utilizado> <parâmetros adicionais do codec> -f <protocolo utilizado para a saída> <endereço ip do destino da stream processada>

Indica-se a seguir os parâmetros utilizados para configurar cada *codec* de vídeo, juntamente com a respetiva justificação:

- **Hevc_qsv**:
    - –preset:v – a biblioteca hevc_qsv possui uma lista de opções que otimizam uma solução de compromisso entre a eficiência de compressão e a velocidade do processo de codificação. Existem 9 predefinições que variam entre *ultrafast* a *veryslow*, sendo que as predefinições mais rápidas resultam em *streams* de vídeo com menor qualidade de imagem e as mais lentas em *streams* de vídeo com maior qualidade. Para o propósito deste trabalho, é aconselhável a utilização de predefinições rápidas (*faster*, *veryfast*) para não criar atrasos na transmissão do vídeo.
    - –load_plugin – as bibliotecas de *software* da Intel necessitam de indicar se o processo é efetuado através de *software* ou de *hardware*, sendo que, neste caso, é necessário indicar a utilização através de *hardware* com a opção *hevc_hw*.

- **Libvpx-vp9**:
    - –speed – este parâmetro regula a velocidade com que a *stream* de vídeo é processada indicando a utilização do CPU com um valor entre 0 e 16, dependendo do que é definido no parâmetro *deadline*.
    - –deadline – este parâmetro é semelhante ao *preset* utilizado no hevc_qsv. Define predefinições que são enquadradas conforme a finalidade da *stream* de vídeo final. As opções disponíveis no VP9 incluem as seguintes: *good*, *best* e *realtime*. *Good* é a predefinição deste parâmetro e é o valor recomendado para a maioria das aplicações; *Best* é a melhor predefinição para uma compressão de vídeo mais eficiente, sacrificando o tempo necessário para concluir o processo; *Realtime* é o valor aconselhado para codificações rápidas para fins que incluem transmissões em tempo real, sacrificando um pouco a qualidade final do vídeo.
    - –tile-columns – esta opção é utilizada juntamente com as duas seguintes. Consoante o valor introduzido (0 a 5), este parâmetro indica ao codificador em quantas regiões retangulares é feita a divisão das *frames*, de forma a processá-las em paralelo para tirar maior partido de sistemas com múltiplos processadores. O valor utilizado neste trabalho é 4 devido aos testes serem executados numa máquina com um processador com as características anteriormente descritas.
    - –frame-parallel – este parâmetro ativa ou desativa (0 ou 1) o processamento das *frames* em paralelo. Neste trabalho foi utilizado o valor 1.
    - –threads – este parâmetro indica o número de *threads* do CPU utilizadas para processar as *frames*. O número é escolhido conforme o número de *threads* disponíveis no processador. O processador utilizado neste trabalho possuí um máximo de 8 *threads*, sendo esse o valor escolhido para este parâmetro.

- **Libaom-av1**:

    - -strict experimental – o AV1 está numa fase inicial de desenvolvimento. Durante esta fase é necessário efetuar a inclusão deste parâmetro de forma a ativar as funcionalidades experimentais disponíveis neste momento, de modo a acelerar o processo de codificação do vídeo.

    - -cpu-used – este valor varia entre 0 e 8, indicando o rácio entre qualidade do vídeo e a velocidade com que este é processado, de forma semelhante ao parâmetro *deadline* da *libvpx-vp9*. Neste trabalho escolheu-se o valor 4 já que este *codec* não é utilizado com a finalidade de *streaming* em tempo real, não existindo, por isso, a necessidade de sacrificar o tempo ou a qualidade.

    - -row-mt – esta opção, quando ativa (1), permite a utilização dos *threads* disponíveis pelo CPU para processar as diferentes regiões das *frames*.

Independentemente do *codec* utilizado, há duas formas de controlar a qualidade final da *stream* de vídeo processada: definindo uma qualidade constante ou definindo um débito binário (*bitrate*) constante. Assim, através do parâmetro *Constant Rate Factor* (-crf) é possível indicar o valor da qualidade de compressão efetuada pelos *codecs*. As escalas do valor escolhido para este parâmetro variam entre 0 e 51 para os *codecs* H.264 e H.265 e entre 0 e 63 para o VP9 e AV1. Quanto maior for o valor escolhido, maior será a compressão obtida, reduzindo-se a qualidade do vídeo. A desvantagem da utilização deste método está relacionada com o fato de se desconhecer o valor da largura de banda necessária para a *stream* processada. É importante relembrar que o objetivo principal deste trabalho é precisamente o de apresentar uma *stream* com um débito binário fixo de modo a não criar flutuações na rede, e assim demonstrar que, com a aplicação de um débito binário mais reduzido, associado à utilização dos *codecs* de nova geração em lugar do *codec* H.264, é possível manter uma qualidade de vídeo muito semelhante à que se obtinha com a simples utilização do H.264. Para fixar o débito binário de saída é necessário modificar os valores dos seguintes parâmetros:

- -b:v/minrate/maxrate – nestes 3 parâmetros indica-se o valor do *bitrate* desejado na *stream* de vídeo final. O valor especificado no *minrate* e no *maxrate* é o mesmo se não for desejável a flutuação da largura de banda durante a

transmissão do vídeo. Se estes dois últimos parâmetros não forem definidos, *"-b:v"* apenas especifica o *bitrate* médio desejado, podendo existir flutuações. Os valores utilizados neste trabalho são definidos mais adiante no Capítulo 5.

- -bufsize – O débito binário introduzido neste parâmetro deve ser, pelo menos, o mesmo valor que foi selecionado para o parâmetro anterior, de modo a permitir ao FFmpeg calcular o tamanho do *buffer* de forma a produzir um débito binário constante. Este parâmetro apenas se utiliza em conjunto com o *maxrate*.

Existem ainda parâmetros adicionais que são relativos a configurações internas do FFmpeg, mas que auxiliam a transmissão da *stream* de vídeo, tais como os seguintes:

- *-max_muxing_queue_size <valor>* - durante o processo de *transcoding* de um media, o FFmpeg não escreve na saída enquanto os pacotes não tiverem sido processados. O valor selecionado para este parâmetro define o tamanho do *buffer* do número de pacotes a armazenar até se iniciar a escrita. O valor 400 foi obtido após experiências efetuadas, tendo-se variado a escolha deste valor até se observar que não existiam erros de transmissão.

- *endereço ip do servidor streaming****?fifo_size=<valor>*** - o *fifo_size* é um *buffer* circular utilizado para o protocolo de rede TCP. Quando o valor predefinido (5 MB) não é suficiente, define-se um valor pretendido conforme o exemplo utilizado. A fórmula utilizada para calcular o valor do *fifo_size* é **valor_em_MB*1024*1024/188**. O valor do *fifo_size* é indicado em Bytes. O valor 188 indica o tamanho em Bytes de 1 pacote.

- *-vf scale=<largura:altura>* - este parâmetro é utilizado em *streams* de vídeo H.264 para efetuar o *downscaling* de forma a comparar a qualidade das *streams* de vídeo codificadas com VP9, HEVC e AV1. Para um vídeo no formato 720p, os valores introduzidos neste parâmetro são 1280:720, correspondendo, respetivamente, ao número de pixéis na horizontal e na vertical.

Durante o processo de transcodificação da *stream* de vídeo, é necessário que a velocidade deste processo seja suficiente de modo a não causar muito impacto relativamente ao tempo de espera nos dispositivos recetores. No FFmpeg a velocidade com que a *stream* de vídeo está a ser processada é medida em fps, indicando o número

de *frames* codificadas por segundo. A Figura 4.9 mostra, no final da linha, a velocidade com que uma determinada *stream* de vídeo está a ser processada, sendo que esse valor deve estar, pelo menos acima, de "1x", de forma a que as fps processadas sejam superiores às fps da *stream* original. O valor exibido é calculado a partir do rácio entre as fps processadas e as fps da *stream* de vídeo original. Se esse valor for abaixo de 1, o T*ranscoder* poderá não dispor de um *buffer* suficientemente largo para armazenar as *frames* antes de proceder ao respetivo envio, podendo, por isso, causar erros de transmissão e prejudicar a respetiva fluidez.

```
frame=  252 fps= 32 q=-0.0 size=    1481kB time=00:00:10.32 bitrate=1175.2kbits/s speed=1.32x
```

Figura 4.9 – Exemplo do processo de *transcoding* a uma velocidade de 1.32x (32 *frames* codificadas por segundo)

4.3.3. Dispositivos

Como já foi referido anteriormente, o módulo de *Transcoder* do Sistema aqui proposto deve conter *chips* gráficos com capacidades de codificar *streams* de vídeo com os recursos disponíveis na plataforma. Este é, de resto, um dos requisitos principais de sistema para efetuar a transcodificação de *streams* de vídeo em tempo real.

O dispositivo utilizado (Figura 4.10) para simular as funcionalidades do *Transcoder* neste projeto é um computador pessoal fabricado pela Intel, com a inclusão de um processador da arquitetura *Skylake*. Esta geração de processadores incluí um processador gráfico com suporte em *hardware* a múltiplos *codecs*, incluído a codificação e descodificação do HEVC, H.264 e apenas a descodificação do VP9. Nos três casos, o formato máximo suportado é 2160p (4K), sendo que no caso do HEVC, a plataforma é capaz de codificar uma *stream* de vídeo até 10 bits de profundidade. As especificações gerais do dispositivo utilizado são identificadas na Tabela 4.1.

Figura 4.10 – Dispositivo utilizado para simular o *Transcoder*

Tabela 4.1 – Especificação do dispositivo escolhido para simular o *Transcoder*

Componente	Valor
CPU	Intel i3-6000U
GPU	Intel HD Graphics 520
RAM	4 GB
Disco	250 GB

É importante referir que este equipamento apenas serve de suporte para o presente trabalho, sendo que não é o mais adequado para suportar a transcodificação de diversas *streams* de vídeo de altas resoluções em simultâneo. A Intel recorre a processadores dedicados a servidores empresariais, e possui, por isso, um maior poder computacional capaz de lidar com processamento gráfico mais intenso. A tabela fornecida pela Intel (Figura 4.11) indica o número de *streams* de vídeo em simultâneo que é atingível por um processador dedicado Intel Xeon E3-1585Lv5.

A Nvidia também possui suporte em *hardware* para a transcodificação de vídeo digital nas respetivas plataformas lançadas recentemente. Juntamente com a biblioteca proprietária NVENC, obtém resultados muito positivos sendo até capaz de codificar *streams* de vídeo com resoluções 8K em plataformas com arquiteturas mais recentes (e.g. Pascal, Volta, Turing). Em ambos os casos, o custo de aquisição destes produtos é

bastante elevado. Naturalmente, o uso destas plataformas deve ser efetuado conforme as finalidades pretendidas.

Video Transcoding Multistream Performance: HEVC on Intel Xeon Processors[1] (1xRT = 30 FPS)		Number of Real-Time Streams (30 FPS)	Number of Real-Time Streams (60 FPS)
1080p to 1080p	AVC to HEVC	15	7
	HEVC to HEVC	8	4
4K to 4K	AVC to HEVC	4	2
	HEVC to HEVC	2	1

Figura 4.11 – Número de *streams* de vídeo simultâneas suportadas num processador dedicado. https://software.intel.com/en-us/intel-media-server-studio

4.4. Recetores

O módulo dos recetores representa a fase final do funcionamento do sistema aqui proposto, tendo como finalidade principal a reprodução da *stream* de vídeo enviada pelo módulo de *Transcoder* através do protocolo de rede UDP. No contexto deste trabalho, os recetores estão divididos em duas categorias distintas de equipamentos: as *TV boxes* e os dispositivos móveis. Ao longo desta secção são indicados os requisitos necessários para cada categoria desses equipamentos.

Os equipamentos televisivos estão geralmente ligados a *codecs* de vídeo relacionados com a norma MPEG devido a todo o seu historial relacionado com as transmissões televisivas, como se reviu na secção 2.2. Atualmente, o *codec* de vídeo utilizado na grande maioria dos meios de transmissão de televisão é o AVC/H.264 devido à sua fiabilidade e estabilidade, servindo o propósito na grande maioria dos casos. Para dar seguimento à reprodução do sinal de vídeo codificado com o *codec* H.264 é necessário que o recetor esteja equipado com um descodificador implementado em *hardware*. Uma das questões principais a ter em conta na utilização deste tipo de equipamentos é a licença de vídeo necessária para implementar o *codec* de vídeo do grupo MPEG. Para melhor ilustrar este aspeto, reproduz-se a seguir um exemplo retirado de um manual de uma *TV box* da Cisco que é utilizada por uma operadora portuguesa, onde se indica

claramente que é necessário que os fornecedores de conteúdos e as emissoras possuam este tipo de licença de vídeo para reproduzirem os conteúdos de forma legal.

"Licença de Vídeo

Relativamente aos produtos AVC/H.264, somos obrigados ao seguinte aviso:

ESTE PRODUTO ESTÁ LICENCIADO SOB A PATENTE AVC PARA USO PESSOAL E NÃO COMERCIAL PARA (i) CODIFICAR VIDEO DE ACORDO COM O STANDARD AVC ("AVC VIDEO") E/OU (ii) DESCODIFICAR VIDEO AVC QUE FOI CODIFICADO POR UM SISTEMA PESSOAL E SEM FINS COMERCIAIS E/OU OBTIDO ATRAVÉS DE UM FORNECEDOR DE CONTEÚDOS LICENCIADO PARA VIDEO AVC. NÃO É FORNECIDA NENHUMA OUTRA LICENÇA, NEM DEVERÁ ESTAR IMPLICITA, PARA QUALQUER OUTRO USO. INFORMAÇÃO ADICIONAL PODE SER OBTIDA JUNTO DA MPEG LA, L.L.C.

CONSULTE http://www.mpegla.com

Assim, informamos que os prestadores de serviços, fornecedores de conteúdo e emissoras são obrigadas a obter uma licença de uso em separado de MPEG LA, antes de qualquer utilização de codificadores AVC/H.264 e / ou descodificadores." (Cisco, 2011)

Ao implementar uma solução baseada no *codec* HEVC para as *TV boxes*, o requisito principal do recetor é a integração do *codec* no equipamento com a indicação da licença de vídeo para permitir a descodificação de conteúdos codificados em HEVC. Além do suporte do *codec* de vídeo, a *TV box* deve possuir uma ligação à Internet com ou sem fios com uma largura de banda suficiente para reproduzir o conteúdo transmitido pelo T*ranscoder*. No projeto descrito nesta dissertação, utiliza-se uma televisão com capacidades de descodificação de vídeo HEVC com uma resolução máxima de 4096 por 2160 pixéis a 60 fps até 8 bits de profundidade e uma ligação à Internet para simular a receção do sinal televisivo por parte de uma *TV box*.

Tal como os equipamentos recetores televisivos, os dispositivos móveis também necessitam de possuir o suporte em *hardware* dos *codecs* de vídeo para permitir a visualização do conteúdo de forma eficiente. As duas plataformas principais no mercado, iOS e Android, possuem políticas um pouco diferentes no suporte dos *codecs* de vídeo. A Apple implementou o HEVC no último trimestre de 2016 juntamente com os seus novos produtos, como foi o caso do iPhone 7. Esta implementação teve como fundamento principal a possibilidade de o utilizador optar por converter a biblioteca de vídeos existente no dispositivo para o formato previsto pela nova norma de codificação

de vídeo de modo a utilizar menos espaço de armazenamento, conservando a mesma qualidade visual. Além disso, o iPhone 7 permite ainda ao utilizador gravar um vídeo diretamente no *codec* HEVC. Em suma, a plataforma iOS para dispositivos móveis está preparada para receber uma *stream* de vídeo codificada em HEVC desde que o equipamento tenha implementado em *hardware* o *codec* em questão. A plataforma Android possui suporte aos *codecs* de nova geração HEVC e VP9 dependendo do fabricante do núcleo de processamento do dispositivo em questão. Uma grande percentagem dos dispositivos móveis Android utiliza atualmente processadores elaborados pela Qualcomm (Qualcomm, 2018a). Inicialmente, a Qualcomm começou por implementar em *hardware* o suporte para o *codec* de vídeo HEVC, sendo que apenas a partir de 2017 é que a fabricante adicionou o suporte ao *codec* VP9. De facto, os dispositivos de gama alta com a plataforma Snapdragon 835, e mais recentemente Snapdragon 845, são os únicos que possuem suporte para ambos os *codecs* de vídeo HEVC e VP9, além dos Snapdragon 660, 670, 675 e 710 em equipamentos de média gama. Os restantes possuem apenas suporte para o *codec* HEVC e H.264 na ausência deste (Qualcomm, 2018b).

Figura 4.12 – App móvel (VLC) utilizada para reproduzir as *streams* de vídeo em dispositivos Android

Neste projeto será utilizado um dispositivo móvel Android da fabricante Huawei que, tal como a Apple, produz os seus próprios processadores. Este dispositivo é equipado

com um processador Kirin 960 com suporte em *hardware* para os *codecs* de vídeo HEVC e H.264 e servirá para comparar a qualidade de imagem das duas *streams* de vídeo codificadas com esses dois *codecs* de vídeo. Para testar a descodificação de *streams* codificadas com o *codec* VP9 é utilizado um computador com suporte em *hardware*.

A app móvel responsável pela reprodução da *stream* de vídeo deve incluir os *codecs* de vídeo necessários para o efeito. Neste projeto utiliza-se a app móvel *VLC for Android* (versão 3.0.13), como se ilustra na Figura 4.12, que incluí suporte para a maioria dos *codecs* existentes, incluindo todos os que são abordados neste trabalho, e permite a reprodução de conteúdos multimédia partilhados em rede.

5. Testes e Resultados

Neste capítulo descreve-se um conjunto de testes que foram realizados com o intuito de avaliar a validade e viabilidade do sistema de transcodificação proposto. Assim, após uma secção inicial em que se descrevem as métricas e ferramentas de comparação entre a qualidade de vídeo, o capítulo prossegue com a descrição dos testes de transcodificação de *streams* de vídeo em tempo real, indicando os resultados obtidos através de métricas de avaliação da qualidade de imagem. Os testes foram realizados com o objetivo principal verificar qual o nível de eficácia da utilização destes *codecs* de vídeo de nova geração, bem como analisar os resultados do processo proposto de transcodificação. As características dos vídeos escolhidos para a realização dos testes que seguem são detalhadas na Tabela 5.1.

Os testes realizados ao *codec* HEVC foram executados no equipamento descrito no Capítulo 4, com suporte em *hardware* para este *codec*. Os *codecs* VP9 e AV1 foram testados num equipamento com maior poder computacional, dado que estes apenas possuem bibliotecas implementadas em *software* no presente trabalho, nomeadamente a *libvpx-vp9* e a *libaom-av1*.

Tabela 5.1 – Descrição das características dos vídeos utilizados nos testes

Título	**Género**	***Frame rate***	**Resolução**	**Profundidade**	***Codec* testado**
Crowd Run	Movimento rápido	50 fps	1920x1080	8 bits	H.264-HEVC
Stockholm	Vista aérea	25 fps	1280x720	8 bits	H.264-VP9
Crosswalk (Netflix)	Movimento lento	25 fps	4096x2160	10 bits	H.264-AV1

5.1. Métricas e ferramentas de comparação entre vídeos

Para finalizar o trabalho e verificar a viabilidade do sistema aqui proposto são necessárias métricas e ferramentas que permitam realizar uma avaliação das diferenças entre as *streams* de vídeo para assim, verificar, até que ponto o sistema de transcodificação proposto funciona de acordo com os objetivos traçados inicialmente.

Os métodos de avaliação aqui utilizados são baseados numa métrica intitulada *Mean Opinion Score* (MOS) que é muito utilizado no domínio da Qualidade da Experiência (QoE) e no setor das telecomunicações. Esta métrica tem como objetivo principal quantificar a perceção, por parte de uma pessoa, da qualidade de media digitais, tais como o vídeo e o áudio. Originalmente, o MOS era calculado a partir da média aritmética da avaliação obtida por pessoas especializadas na matéria, produzindo um valor aproximado à décima numa escala de um a cinco, em que um representa "mau" e cinco "excelente". O valor do MOS é determinado através da seguinte expressão:

$$MOS = \frac{\sum_{n=1}^{N} Rn}{N}$$

em que **R** representa a avaliação proporcionada pelos participantes individuais e **N** representa o número total de avaliações.

O método de avaliação aqui proposto é relevante para este propósito já que serão as pessoas os consumidores finais dos objetos avaliados. No entanto, os testes realizados possuem desvantagens tais como o tempo e os recursos necessários para realizar essas tarefas, além dos resultados não serem objetivos por resultarem de perceções subjetivas proporcionadas por seres humanos.

O sistema visual humano possui fraquezas e limitações, tal como foi explicado na secção 2.3. Esta limitações fazem com que não se possa utilizar apenas a visualização dos vídeos como método de avaliação e comparação entre vídeos semelhantes, sendo necessários algoritmos dedicados para realizarem essas tarefas. Atualmente as avaliações efetuadas a conteúdos digitais não são realizadas através de um grupo de pessoas qualificadas mas através de algoritmos dedicados com o objetivo de aproximar a experiência obtida por parte de um ser humano. Esses algoritmos estão distribuídos por duas categorias fundamentais de métricas de qualidade de vídeo:

- Métricas sem referência (*No-reference metrics*) – são algoritmos que indicam a qualidade de um determinado vídeo sem recorrer a uma comparação com outro vídeo. Medem a qualidade de um formato de vídeo em concreto para obter os resultados de distorções causados pela codificação ou transmissão do vídeo. Isto implica que o algoritmo tenha conhecimento de quais os diferentes tipos de distorções causados pelos diferentes *codecs* de vídeo para efetuar a medição.

- Métricas de referência completa (*Full-reference metrics*) – estes tipos de métricas incluem dois vídeos de *input* para avaliar, sendo que um é o vídeo original que serve de referência, e o segundo é o mesmo vídeo processado após a transcodificação. Estes algoritmos permitem uma avaliação do desempenho dos codificadores de forma mais precisa, colocando o vídeo de referência no *input* do codificador e, como resultado, o vídeo avaliado. Isto permite quantificar as diferenças obtidas entre os vídeos testados obtendo um valor que indica a diferença entre ambos.

Neste projeto utilizam-se as métricas de referência completa devido ao interesse em adquirir as diferenças entre a *stream* de vídeo proveniente do Servidor de *streaming* de vídeo e a *stream* de vídeo resultante da transcodificação. Detalha-se a seguir a lista das métricas que são utilizadas nos testes realizados neste capítulo, juntamente com a respetiva definição.

As métricas utilizadas nos testes descritos neste capítulo incluem as seguintes:

- **PSNR (*Peak Signal-to-Noise Ratio*);**

- **SSIM (*Structural Similarity*);**

- **VMAF (*Video Multi-Method Assessment Fusion*);**

- **VQM (*Video Quality Model*).**

A métrica **PSNR** (*Peak Signal-to-Noise Ratio*) é uma das métricas mais utilizadas em avaliações de imagem e vídeo. O seu resultado é determinado numa escala logarítmica medida em *decibel* e calcula-se através seguinte expressão:

$$PSNR(Img1, Img2) = 10log_{10}\frac{(2^n - 1)^2}{MSE(Img1, Img2)}$$

onde **MSE** (*Mean Squared Error*) é a média quadrática da diferença entre duas imagens (sendo que uma é a original e a outra a processada), e **n** é o número de bits necessários para representar a imagem.

Este algoritmo quantifica a diferença absoluta entre dois sinais idênticos, comparando *byte* a *byte* essas diferenças sem considerar o verdadeiro significado desses dados. O PSNR ignora por completo os componentes do vídeo tais como os pixéis e a relação entre eles, fazendo com que seja a ferramenta ideal para diferenciar de forma detalhada dois sinais de vídeo, mas não tão conveniente para simular as diferenças captadas pelo sistema visual humano, muito devido a apenas medir a componente *luma* da imagem. Para completar essa lacuna, foi criada uma extensão do PSNR intitulada de PSNR-HVS que incorpora certas propriedades do sistema visual humano (HVS) como a perceção de diferenças de contraste de uma imagem. Tipicamente, os valores do PSNR na compressão de imagens e vídeo variam entre 30 dB e 50dB, sendo que valores mais elevados indicam uma maior semelhança entre a imagem original e a imagem processada (neste caso, comprimida e enviada por *streaming*).

Por outro lado, a **SSIM** (*Structural Similarity*), tal como o PSNR, é uma métrica para avaliar as diferenças percetuais entre imagens e *frames* individuais entre um vídeo de referência e um vídeo processado. A diferença principal deste método para o anterior é que o método SSIM tem como objetivo fazer a medição das semelhanças e da degradação ocorrida nas estruturas da imagem processada em relação à imagem original, enquanto que o PSNR estima os erros absolutos *byte* a *byte* dessa mesma imagem. Uma das variantes deste método é o *Multi-Scale SSIM* (MS-SSIM) que obtém um desempenho idêntico ou superior em relação ao SSIM , obtendo diferentes escalas para diferentes setores da imagem. A escala utilizada neste tipo de métricas é, geralmente, arredondada às centésimas, e varia entre 0, que representa o pior valor possível, e 1, que representa o melhor valor possível e que só é atingível ao comparar a mesma imagem original, sendo que um valor decimal muito próximo de 1 indica que a imagem processada obtém um nível alto de semelhança em comparação com a imagem original.

Por sua vez, o método **VMAF** (*Video Multi-Method Assessment Fusion*)[15] foi desenvolvido na Universidade da Califórnia do Sul com o intuito de elaborar uma métrica resultante de um conjunto de diferentes métricas para avaliar a qualidade entre as imagens originais e as imagens processadas e também certas características do vídeo digital. Este método foi concretamente concebido para a Netflix para avaliar o desempenho dos respetivos vídeos com o objetivo de transmitir conteúdos com melhor qualidade. Como cada métrica geralmente possui as suas vantagens e desvantagens, e diferentes finalidades, o VMAF pretende efetuar uma fusão dessas métricas através de algoritmos de *machine-learning* que atribuem um peso de forma dinâmica a cada métrica, obtendo um resultado final mais fidedigno. As métricas utilizadas no VMAF são o *Visual Information Fidelity* (VIF) e o *Detail Loss Metric* (DLM) para as imagens e uma ferramenta que visa explorar as características temporais do vídeo (*motion*) entre *frames* (Li, Anne, Katsavounidis, Moorthy, & Manohara, 2016). A escala utilizada neste método varia de 0 a 1, sendo que quanto maior for o valor resultante, melhor será a qualidade final do vídeo.

Finalmente, o método **VQM** (*Video Quality Model*) foi desenvolvido pelo *Institute for Telecommunication Sciences* (ITS) com o objetivo de medir a qualidade do vídeo digital. Esta avaliação prevê uma semelhança em relação à avaliação subjetiva obtida por um conjunto de visualizadores humanos. O VQM é considerado pelo *International Video Quality Expert's Group* (VQEG) como uma norma para avaliar o desempenho de vídeos digitais, sendo até adotado nas normas ANSI e do ITU. Uma das grandes vantagens do VQM é ser livre de pagamento de *royalties*, podendo ser utilizado de forma comercial e não comercial (Institute for Telecommunication Sciences (ITS), 2018).

Algumas destas métricas possuem ferramentas de *software* associadas que foram concebidas pelo mesmo grupo que desenvolveu a métrica, como é o caso do VMAF ou VQM, mas, na generalidade, existem programas que incluem estas métricas e que possibilitam gerar dados estatísticos com base nos resultados obtidos. Identifica-se a seguir uma lista dos programas utilizados neste projeto para concretizar a avaliação dos vídeos utilizados, juntamente com as métricas suportadas por cada um deles.

[15] https://github.com/Netflix/vmaf

As ferramentas de *software* que aplicam as métricas indicadas anteriormente incluem as seguintes:

- a **VQMT**[16] (*Video Quality Measurement Tool*): desenvolvido pelo EPFL (Lausanne, Suíça), permite avaliar vídeos com as seguintes métricas:
    - PSNR (PSNR-HVS)
    - SSIM (MS-SSIM)

A ***MSU Video Quality Tool***[17]**:** permite avaliar vídeos com todas as quatro métricas acima referidas.

5.2. Realização de testes

5.2.1. HEVC

O sistema foi testado primeiramente com o *codec* HEVC, de forma a utilizar as características do equipamento com suporte em *hardware* deste *codec*. Foram efetuadas quatro transmissões com quatro débitos binários de referência (1500 Kbps, 2500 Kbps, 4000 Kbps e 6000 Kbps) para analisar as diferentes qualidades obtidas após a transcodificação das *streams* H.264 e H.265. Para que se obtivesse uma comparação justa, nestes testes efetuados sobre o *codec* H.265 foi utilizada a biblioteca *h264_qsv* para as *streams* transcodificadas em H.264, de forma a comparar as duas API que implementam comandos em *hardware*. O processo de transcodificação das *streams* de vídeo em ambos os *codecs* de vídeo ocorreu de forma fluída, mantendo as velocidades acima de 1 independentemente do débito binário utilizado, conforme indicam os resultados apresentados nas Figura 5.1 e 5.2.

```
      cpb: bitrate max/min/avg: 6000000/0/6000000 buffer size: 12000000 vbv_delay: -1
frame=  500 fps=123 q=-1.0 Lsize=    7758kB time=00:00:09.98 bitrate=6367.6kbits/s speed=2.46x
video:7755kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: 0.036986%
```

Figura 5.1 – Velocidade da transcodificação da *stream* H264 (2.46x) a 6000 kbps

```
      cpb: bitrate max/min/avg: 6000000/0/6000000 buffer size: 12000000 vbv_delay: -1
frame=  500 fps= 66 q=-1.0 Lsize=    7511kB time=00:00:09.94 bitrate=6189.8kbits/s speed=1.32x
video:7504kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: 0.090475%
```

Figura 5.2 - Velocidade da transcodificação da *stream* H265 (1.32x) a 6000 kbps

[16] https://mmspg.epfl.ch/vqmt
[17] http://www.compression.ru/video/quality_measure/video_measurement_tool.html

A sequência de vídeo *Crowdrun* possui um débito binário original de 38 Mbps e é, do ponto de vista computacional, muito exigente para os codificadores de vídeo, dado que contém muitas pessoas a correrem, aumentado a imprevisibilidade das imagens. Para além do fator da imprevisibilidade, o vídeo foi reproduzido a 50 fps, sendo, por isso, outro fator de dificuldade adicional.

Outro processo que exige um elevado poder computacional é a obtenção dos resultados de qualidade dos vídeos processados. Em média são necessários 70 segundos para completar as análises com as métricas MS-SSIM e PSNR-HVS para analisar 100 *frames* da sequência de vídeo em questão. A Figura 5.3 ilustra os resultados obtidos, indicando o tempo que demorou (em segundos).

```
C:\Users\Joel\Desktop\videos y4m>VQMT.exe .\original\crowdrun_1080_original.y4m .\h264\crowdrun\crowdrun_x264
_6000.y4m 1920 1072 100 1 .\h264\crowdrun\crowdrun_6000 MSSSIM PSNRHVS
Time: 70.172s
```

Figura 5.3 – Obtenção dos resultados das métricas MS-SSIM PSNRHVS para o vídeo *Crowdrun*

5.2.2. VP9

Os testes realizados ao *codec* VP9 foram efetuados num computador com um maior poder computacional dada a inexistência, neste projeto, de equipamentos com suporte em *hardware* deste *codec*. As bibliotecas de *software* de referência utilizadas foram a *libvpx-vp9* e a *libx264* que implementam, respetivamente, os *codecs* VP9 e H.264 em *software*. Tal como foi efetuado nos testes do HEVC, os débitos binários utilizados para esta sequência de vídeo são adequados à resolução do vídeo (1280x720 pixéis), sendo de 1000 Kbps, 1500 Kbps, 2500 Kbps e 4000 Kbps. Apesar dos testes terem ocorrido num ambiente pouco propício para transcodificar *streams* de vídeo em tempo real, as velocidades obtidas durante o processamento dos vídeos foram muito satisfatórias, como se pode observar nos resultados ilustrados na Figura 5.4 e na Figura 5.5. Estes resultados foram conseguidos devido ao facto de se ter utilizado um computador com vários núcleos de processamento, e também pelo facto do vídeo possuir uma resolução pouco elevada, sendo que a dificuldade de processamento é maior quanto maior for a resolução do vídeo a processar.

```
      cpb: bitrate max/min/avg: 4000000/4000000/4000000 buffer size: 0 vbv_delay: -1
frame=  254 fps= 54 q=0.0 Lsize=    5276kB time=00:00:10.12 bitrate=4270.2kbits/s speed=2.15x
video:5273kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: 0.052339%
```

Figura 5.4 – Velocidade da transcodificação da *stream* VP9 (2.15x) a 4000 kbps

```
      cpb: bitrate max/min/avg: 4000000/0/4000000 buffer size: 8000000 vbv_delay: -1
frame=  254 fps=146 q=-1.0 Lsize=    5351kB time=00:00:10.12 bitrate=4331.8kbits/s speed=5.82x
video:5349kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: 0.035269%
```

Figura 5.5 – Velocidade da transcodificação da *stream* H.264 (5.82x) a 4000 kbps

A sequência de vídeo utilizada nestes testes corresponde a uma vista aérea que capta a cidade de Estocolmo. A câmara utilizada neste vídeo movimenta-se apenas na horizontal, da esquerda para a direita, captando alguns movimentos ocorridos por pessoas e automóveis. Este vídeo permite testar vários aspetos dos *codecs* tais como a predição de movimentos e a forma como lidam com padrões repetidos como os que ocorrem no céu. O débito binário desta sequência (10.7 Mbps) é inferior ao do exemplo apresentado nos testes do HEVC devido à resolução de vídeo ser inferior e à menor imprevisibilidade. Devido a estes fatores, a duração das medições efetuadas às métricas também foi inferior, demorando, em média, cerca de 25 segundos a obter-se os resultados finais.

5.3. Discussão dos resultados

Após a realização destes testes, foi possível analisar e efetuar comparações entre as diferentes qualidades obtidas pelas *streams* de vídeo transmitidas. O objetivo principal deste trabalho é verificar se é possível utilizar *codecs* de vídeo mais recentes e alcançar uma qualidade de imagem semelhante ou superior à que é proporcionada pelo *codec* H.264 utilizando o mesmo débito binário. A Figura 5.6 representa uma *frame* codificada em H.264 a um *bitrate* de 6000 Kbps e, abaixo, a mesma *frame* codificada em H.265 a um *bitrate* inferior (4000 Kbps). Mesmo sem a utilização de algoritmos que implementam métricas de avaliação de qualidade de vídeo é possível observar-se diretamente por inspeção as melhorias no segundo vídeo. Na zona contornada a vermelho é possível observar-se a degradação da qualidade no vídeo codificado em H.264. Pelo contrário, e com um débito binário inferior, na imagem abaixo observa-se uma melhoria significativa da qualidade da mesma região da imagem face à imagem codificada em H.264.

Figura 5.6 – Vídeo H.264 a 6000 kbps (cima) vs Vídeo HEVC a 4000 kbps (baixo)

Para comprovar de forma detalhada as diferenças entre as qualidades apresentadas pelas *streams* de vídeo, foram elaborados gráficos que mostram os valores das métricas calculadas para os diferentes débitos binários de cada vídeo.

Nos testes realizados com o VP9, é possível observar-se na Figura 5.7 e na Figura 5.8 que o VP9 apresenta um melhor desempenho na qualidade de vídeo quando comparado com o H.264. A baixa complexidade de cálculo durante a codificação das *streams* de vídeo favorece o desempenho de ambos os *codecs*, sendo que o VP9, neste caso, consegue ser superior. O segmento de vídeo apresentado possui uma qualidade superior com o VP9 em débitos binários inferiores do que com débitos binários superiores com o H.264. A métrica VQM também indica uma melhoria substancial mediante a utilização do VP9, como se mostra na Figura 5.9.

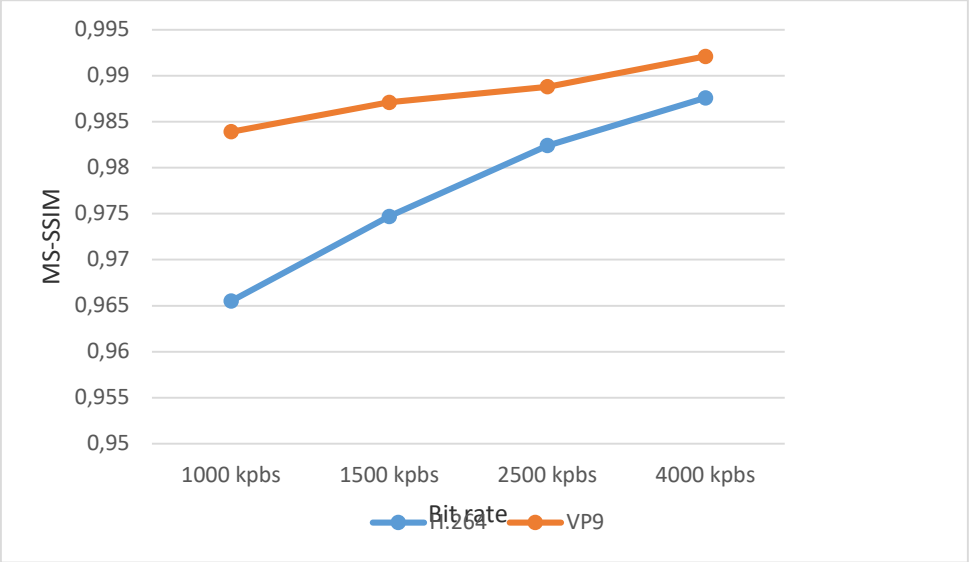

Figura 5.7 – Métrica MS-SSIM calculada para *streams* VP9

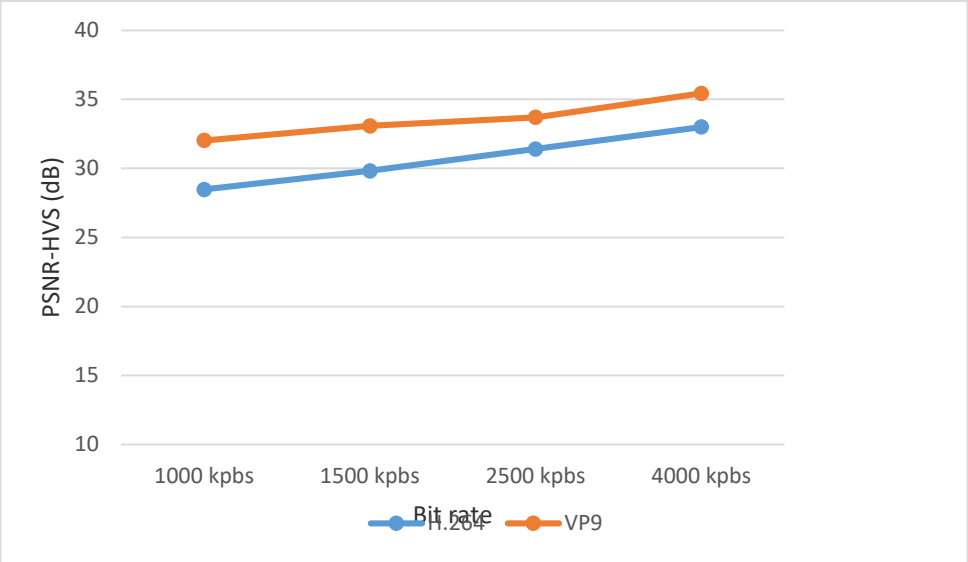

Figura 5.8 – Métrica PSNR-HVS calculada para *streams* VP9

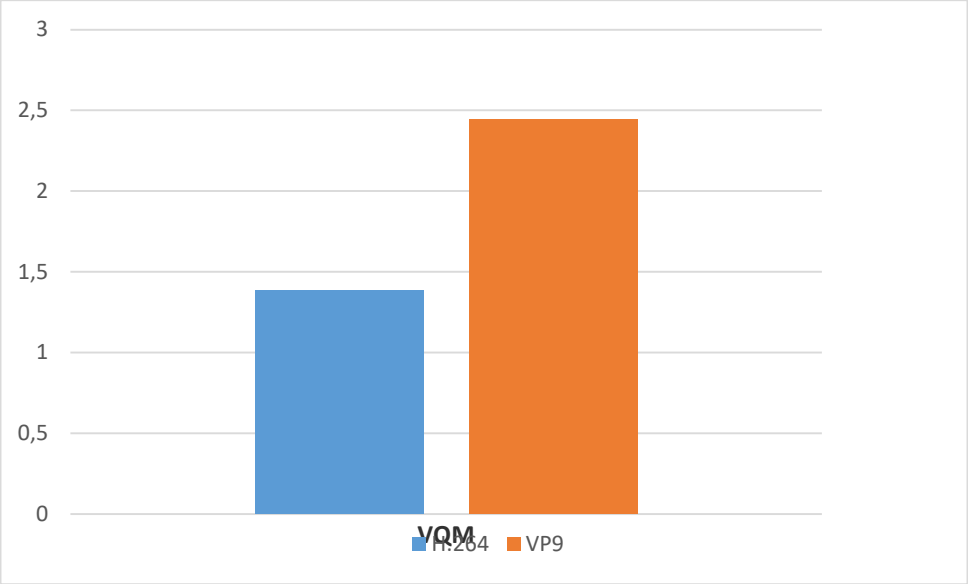

Figura 5.9 – Métrica VQM calculada entre *streams* H.264 e VP9 (1000 Kbps)

Os testes efetuados com o HEVC obtiveram resultados semelhantes aos testes que foram efetuados com o VP9, apontando para a conclusão que o *codec* mais recente obtém resultados mais positivos quando comparado com o *codec* H.264. A Figura 5.10 e a Figura 5.11 mostram os resultados obtidos pelas métricas MS-SSIM e PSNR-HVS, indicando uma melhoria de qualidade de vídeo em débitos binários superiores, sendo que o HEVC supera na qualidade em todos os débitos binários testados, com exceção da métrica PSNR-HVS na *stream* de vídeo de 1500 Kbps. Devido à maior complexidade

ocorrida no vídeo, os valores são ligeiramente inferiores aos que foram observados no segmento de vídeo apresentado anteriormente nos testes realizados ao VP9.

Em ambos os *codecs* testados fica assim revelada a eficácia da aplicação destes métodos, o que nos permite concluir que é possível transmitir conteúdos com qualidade em tempo real, reduzindo a largura de banda necessária para realizar a operação, de acordo com o objetivo principal que foi definido para este trabalho.

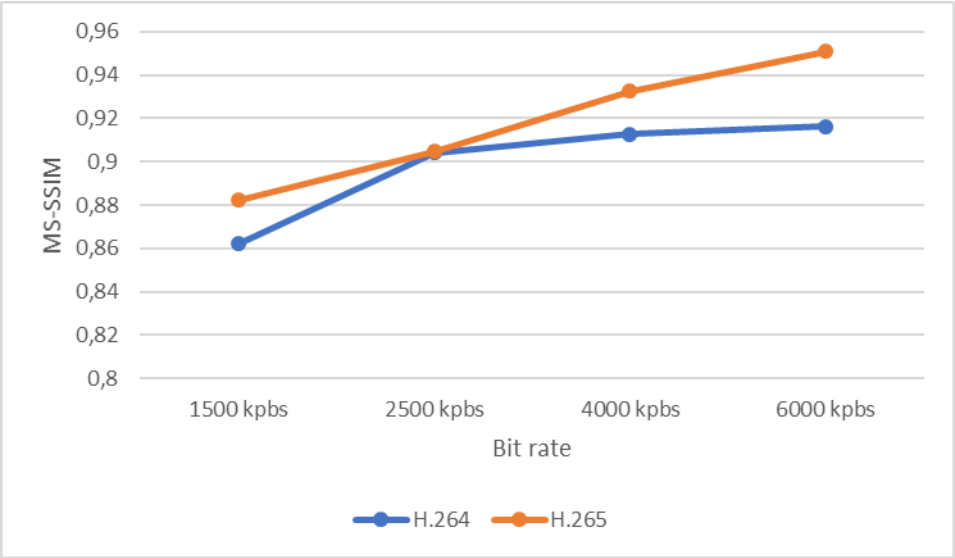

Figura 5.10 - Métrica MS-SSIM calculada para *streams* HEVC

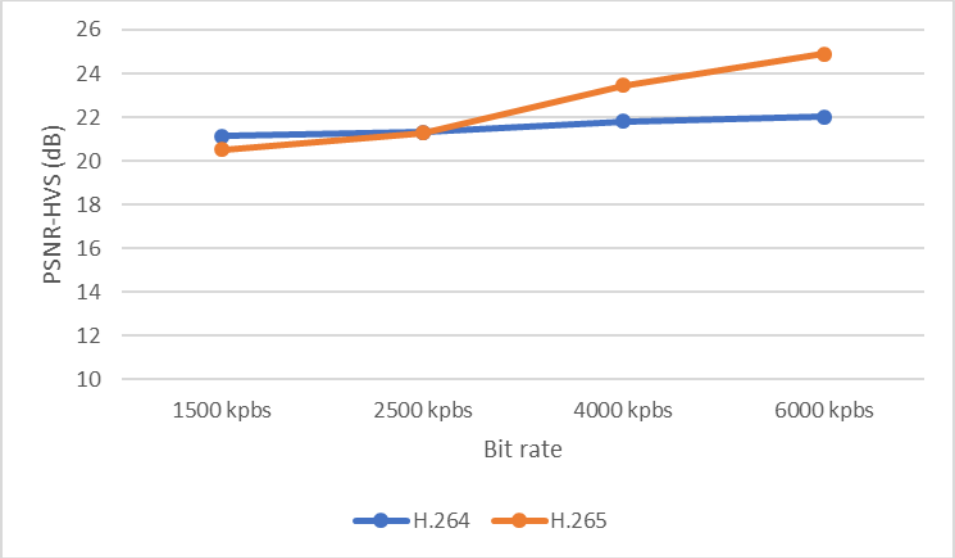

Figura 5.11 - Métrica PSNR-HVS calculada para *streams* HEVC

5.4. O caso do AV1

O AV1, como já foi explicado ao longo deste trabalho, é um *codec* de vídeo que está numa fase inicial de desenvolvimento. Isto significa que, atualmente, este *codec* não pode ser utilizado para efeitos de *streaming* em tempo real em nenhuma circunstância, dado o tempo elevado de que ainda necessita para codificar um vídeo. Para testar o *codec* foi utilizado um segmento de vídeo de cinco segundos, gravado pela Netflix numa resolução 4K com 10 bits de profundidade, juntamente com a biblioteca de *software libaom-av1*. A Figura 5.12 indica que o AV1 codificou o vídeo original a uma velocidade de 0.00216x. Com este valor é possível obter o tempo em segundos que o *codec* necessitou para completar o processo efetuando o seguinte cálculo: 5 / 0.00216 = 2314 segundos, isto é, cerca de 38 minutos para codificar um segmento de vídeo de apenas 5 segundos.

```
      cpb: bitrate max/min/avg: 0/0/0 buffer size: 0 vbv_delay: -1
frame=  127 fps=0.1 q=0.0 Lsize=    6125kB time=00:00:05.04 bitrate=9953.1kbits/s speed=0.00216x
video:6123kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: 0.025470%
```

Figura 5.12 – Codificação em AV1 a uma velocidade de 0.00216x

Para além do tempo necessário para codificar a *stream*, o AV1 utiliza recursos excessivos a nível computacional, nomeadamente uma grande quantidade de memória RAM para efetuar a operação de conversão de formato. A Figura 5.13 demonstra a utilização de quase 6 GB pelo processo do FFmpeg durante a codificação do segmento de vídeo com o *codec* AV1. Estes dados indicam uma eficiência ainda muito escassa no processo de codificação com o AV1, sendo inconcebível, para já, a sua implementação num sistema de *streaming* atual.

Nome	PID	Estado	Nome de ut...	CPU	Memória (c...
ffmpeg.exe	11808	Em execução	Joel	96	5 705 428 K

Figura 5.13 – Utilização de memória RAM do FFmpeg com a *libaom-av1*

Apesar do mau desempenho do AV1 a nível temporal na codificação de vídeos, este *codec* apresenta resultados muito favoráveis no que diz respeito a melhorias de qualidade de vídeo quando comparado com o H.264. De facto, na Figura 5.14 é possível visualizar os resultados obtidos pela métrica MS-SSIM em dois vídeos codificados pelos *codecs* H.264 e AV1. O vídeo codificado em AV1 apresenta uma melhoria de qualidade de imagem face ao H.264 com um débito binário inferior. Isto indica o bom

funcionamento do AV1 em sequências de vídeo com uma resolução elevada, mesmo estando ainda numa fase inicial de desenvolvimento.

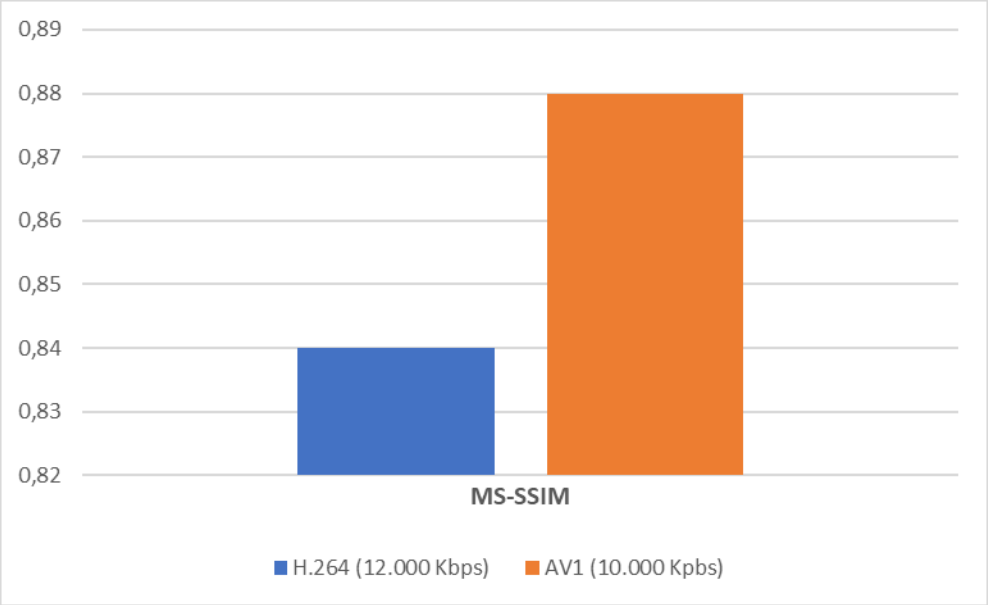

Figura 5.14 – Comparação da métrica MS-SSIM entre H.264 (12.000 Kbps) e AV1 (10.000 Kbps)

6. Conclusão

Este trabalho pretendeu demonstrar a viabilidade da utilização dos *codecs* de nova geração para a transcodificação de *streams* de vídeo, permitindo reduzir a largura de banda necessária para a transmissão, mantendo uma qualidade visual semelhante à *stream* de vídeo codificada em H.264, a norma de referência que é utilizada atualmente, tendo, neste contexto, sido definidos três objetivos principais.

A revisão bibliográfica aqui apresentada permitiu efetuar uma análise aprofundada sobre várias características, métodos e os algoritmos mais atuais para a codificação de vídeo digital. Efetuou-se ainda uma revisão sobre a utilização do vídeo digital em duas áreas tecnológicas proeminentes na atualidade: na Web e no contexto das transmissões de televisão. Deu-se particular relevância à revisão dos tópicos relacionados com a área da televisão devido a ser a área em que este projeto se pretendeu enquadrar. Após a revisão sobre a utilização do vídeo digital, efetuou-se igualmente uma análise mais aprofundada a cada *codec* de vídeo, indicando as suas características principais e as melhorias que introduziu quando comparado com os seus antecessores, cumprindo assim o segundo objetivo estipulado para este trabalho.

O Capítulo 3 descreveu a conceção de um Sistema de *streaming* de vídeo constituído por três módulos principais cujo funcionamento foi descrito e, posteriormente, utilizado para implementar um protótipo que permitisse realizar os testes previstos no contexto deste projeto. Assim, descreveu-se a arquitetura do Sistema especificando os componentes que constituem cada um dos respetivos módulos bem como as respetivas funcionalidades. O Servidor de *streaming* de vídeo apresenta características semelhantes ao que pode ser produzido num estúdio televiso, indicando quais os tipos de vídeos transmitidos pelo servidor e as características que possuí. O módulo de *Transcoder* é o módulo principal deste projeto, e é capaz de processar as *streams* de vídeo transmitidas pelo Servidor de *streaming* de forma a disponibilizar uma qualidade semelhante ao vídeo original com uma menor utilização de largura de banda. Dada a sua importância, definiu-se um conjunto de requisitos para o *Transcoder*, tais como o suporte em *hardware* dos *codecs* abrangidos neste trabalho, com a exceção do AV1. Os recetores atuam como atores neste projeto, sendo os dispositivos responsáveis pela

receção do produto final transmitido pelo *Transcoder*. Neste módulo foram assinalados os diferentes tipos de dispositivos capazes de receber as *streams* de vídeo, indicando as respetivas propriedades e características aconselháveis para uma reprodução fluída do vídeo. Finalizou-se a especificação do Sistema com uma contextualização do utilizador, onde se apresentaram as condições em que se encontra o utilizador final, as motivações para utilizar este tipo de plataforma, os requisitos a que o sistema deve obedecer e, finalmente, os casos de uso que exemplificam o respetivo funcionamento.

Posteriormente efetuou-se uma análise detalhada do processo de implementação de cada módulo do sistema num protótipo funcional que permitisse a realização dos testes. Em cada tópico apresentou-se as ferramentas de *software* utilizadas para o efeito, juntamente com o *hardware* de teste selecionado para este projeto, bem como as características de outros equipamentos mais adequados para a obtenção de melhores resultados. Deu-se uma ênfase particular ao FFmpeg dado que é a ferramenta mais comum que existe quer no Servidor de *streaming* quer no *Transcoder*. Especificou-se a utilidade do FFmpeg em ambos os módulos bem como as finalidades com que é utilizado, indicando os parâmetros e as configurações necessárias a efetuar para o funcionamento correto de cada um dos dois módulos. Juntamente com o FFmpeg foram apresentadas as API utilizadas para a codificação das *streams* de vídeo por *software* e *hardware*.

A seguir apresentou-se um conjunto de testes realizados a um conjunto de diferentes tipos de vídeos com o objetivo de fundamentar a utilização dos *codecs* de vídeo HEVC e VP9. Inicialmente, detalhou-se um conjunto de métricas indicadas para avaliar os resultados finais obtidos pelo *Transcoder* assim como a respetiva justificação e as ferramentas de *software* necessárias para o respetivo uso. Os resultados dos testes foram obtidos através da aplicação das métricas implementadas pelas ferramentas de *software* indicadas. Os resultados obtidos desta avaliação permitem concluir que é possível transmitir conteúdos com qualidade em tempo real, reduzindo a largura de banda necessária para realizar a operação, de acordo com o objetivo principal que foi definido para este trabalho. Por isso, é igualmente possível concluir que um processo de *transcoding* tal como o que aqui se propõe é viável para minimizar os recursos financeiros da transição para um novo *codec* cumprindo assim o primeiro objetivo definido para este trabalho.

Para além da avaliação de ambos os *codecs* de vídeo, fez-se ainda uma avaliação e breve análise do *codec* AV1, não estando muito completa devido ao facto de se tratar ainda de um *codec* em desenvolvimento com perspetivas de se vir a afirmar como uma nova norma de vídeo digital num futuro próximo. No final conclui-se que os *codecs* de vídeo HEVC e VP9 revelaram resultados positivos no processo de *transcoding*, demonstrando que foi possível transmitir *streams* de vídeo com qualidade em tempo real, reduzindo a largura de banda necessária para efetuar a operação, alcançando deste modo o terceiro objetivo definido para este trabalho.

6.1. Trabalho Futuro

Apesar de funcional, o sistema implementado no presente trabalho carece de melhorias significativas de forma a melhorar o desempenho no processo de transcodificação, bem como o número de *streams* de vídeo permitidas em simultâneo. A utilização de equipamentos com maior poder computacional abre o caminho a uma maior diversidade de vídeos para testar, incluindo vídeos de maior resolução e maior profundidade de imagem.

Outro aspeto consideração considerar para trabalho futuro é a criação de uma ferramenta de *software* que implemente a técnica MPEG-DASH para realizar a gestão das *streams* de vídeo nos recetores. Atualmente, estes protocolos são utilizados em plataformas de vídeo digital para adaptar o débito binário das *streams* de vídeo à largura de banda disponível num determinado momento. O objetivo principal seria o de implementar uma ferramenta de *software* que, além de adaptar o débito binário, também adaptasse de forma dinâmica o *codec* utilizado, para assim maximizar a qualidade de vídeo para uma determinada largura de banda.

Por fim, está prevista uma investigação mais detalhada sobre o *codec* AV1 de forma a que, num futuro próximo, seja possível incluir este *codec* em sistemas semelhantes ao que foi desenvolvido neste projeto. Como se afirmou ao longo do trabalho, o AV1 é um *codec* de vídeo com muito potencial para se tornar uma norma internacional nos próximos anos. A implementação deste *codec* num trabalho futuro seria uma mais-valia para ampliar as funcionalidades desenvolvidas para este projeto

Referências

Andrey Norkin, Jan De Cock, Aditya Mavlankar, A. A. (2016). More Efficient Mobile Encodes for Netflix Downloads. Retrieved from https://medium.com/netflix-techblog/more-efficient-mobile-encodes-for-netflix-downloads-625d7b082909

AOM. (2015). Alliance for Open Media Established to Deliver Next-Generation Open Media Formats. Retrieved from https://aomedia.org/alliance-to-deliver-next-generation-open-media-formats/

AOM. (2016). The Alliance for Open Media Welcomes New Members and Announces Availability of Open Source Video *Codec* Project. Retrieved from https://aomedia.org/the-alliance-for-open-media-welcomes-new-members-and-announces/

Apple. (2018). Utilizar conteúdos multimédia com HEIF ou HEVC em dispositivos Apple. Retrieved from https://support.apple.com/pt-pt/HT207022

Bing, B. (2015). *Next Generation Video Coding and Streaming*. Wiley.

C. Antoniou, Z. (2017). *Real-Time Adaptation to Time-Varying Constraints for Reliable mHealth Video Communications*. University of Cyprus. Retrieved from https://www.researchgate.net/figure/Block-Partitioning-in-HEVC-Example-based-on-57-Largest-Coding-Unit-LCU-Coding-tree_fig2_322523982

Chen, Y., Murherjee, D., Han, J., Grange, A., Xu, Y., Liu, Z., … Bankoski, J. (2018). An Overview of Core Coding Tools in the AV1 Video *Codec*. *Picture Coding Symposium (PCS)*, 5. Retrieved from https://ieeexplore.ieee.org/document/8456249

Cisco. (2011). Cisco ISB2231. Retrieved from https://conteudos.meo.pt/meo/Documentos/Manuais/Box/MEO-Fibra-ADSL/Manual-Cisco-ISB2231.pdf?_ga=2.257160587.181538058.1547144912-2119096334.1547144912

Cisco. (2015). Cisco Visual Networking Index: Forecast and Methodology, 2015-2020. *Forecast and Methodology*, 22. https://doi.org/1465272001663118

Dalal, M., & Juneja, M. (2018). International Journal of Computer Sciences and Engineering Open Access H . 264 / AVC Video Steganography Techniques : An Overview, (August). https://doi.org/10.26438/ijcse/v6i5.297303

Dataset, M. D., Feldmann, C., Timmerer, C., & Klagenfurt, A. (2018). Multi-*Codec* DASH Dataset.

DAVIS, A. (2017). Did You Know? The First Live Television Broadcast Demonstration Was in 1926. Retrieved from http://theinstitute.ieee.org/tech-history/technology-history/did-you-know-the-first-live-television-broadcast-demonstration-was-in-1926

DVB.org. (2013). CSA Recommends DVB-T2. Retrieved from https://www.dvb.org/news/csa-recommends-dvb-t2/country/france

EBU-UER. (2010a). *High Definition (HD) Image Formats for Television Production*. Geneva.

EBU-UER. (2010b). *Information Paper on HDTV Formats*.

Facebook. (2018). AV1 beats x264 and libvpx-vp9 in practical use case. Retrieved from https://code.facebook.com/posts/253852078523394/av1-beats-x264-and-libvpx-vp9-in-practical-use-case

FFmpeg. (2018). HWAccelIntro. Retrieved from https://trac.ffmpeg.org/wiki/HWAccelIntro

Ha, L. (2018). *The Audience and Business of YouTube and Online Videos*. Lexington Books. Retrieved from https://books.google.pt/books?id=UoNaDwAAQBAJ

HEVC Advance LLC. (2016). HEVC Advance Announces “Royalty Free” HEVC Software. *PR Newswire*. Retrieved from https://www.prnewswire.com/news-releases/hevc-advance-announces-royalty-free-hevc-software-300367212.html

HEVC Advance LLC. (2018). *Royalty Rate Structure for Trademark Licensees In-*

Compliance Royalty Rate Structure for Trademark Licensees In-Compliance , Effective January 1 , 2018. Retrieved from https://www.hevcadvance.com/pdfnew/RoyaltyRatesSummary.pdf

Institute for Telecommunication Sciences (ITS). (2018). VQM Frequently Asked Questions (FAQ). Retrieved from https://www.its.bldrdoc.gov/resources/video-quality-research/vqm-faq.aspx

Intel Corporation. (n.d.). White Paper : Intel® Quick Sync Video and FFmpeg Installation and Validation Guide. *0 BioRegulations*, *8*(6), 715–721. https://doi.org/10.1016/j.jacr.2011.01.011

KRIEGER, J. (n.d.). DVB-T2 to launch in Germany with 40 channels. Retrieved from https://www.broadbandtvnews.com/2016/06/06/dvb-t2-to-launch-in-germany-with-40-channels/

Larabel, M. (2018). Google Starts Pushing Out VP10 Open-Source Code Into LibvpxNo Title. Retrieved from https://phoronix.com/scan.php?page=news_item&px=Libvpx-VP10-Starts

Lederer, S., & Lederer, S. (2017). Video Developer Report 2017.

Li, Z., Anne, A., Katsavounidis, I., Moorthy, A., & Manohara, M. (2016). Toward A Practical Perceptual Video Quality Metric. Retrieved from http://techblog.netflix.com/2016/06/toward-practical-perceptual-video.html

LionDoc. (n.d.). RGB to Y'CrCb conversation. Retrieved from https://commons.wikimedia.org/wiki/File:CCD.png

Massimino, P. (2017). AOM - AV1 How does it work ?, (July).

Mukherjee, D., Han, J., Bankoski, J., & S Bultje, R. (2015). A Technical Overview of VP9 : The latest royalty-free video *codec* from Google Outline. *SMPTE Motion Imaging Journal*, 124. Retrieved from https://www.researchgate.net/publication/272399193_A_Technical_Overview_of_VP9--the_Latest_Open-Source_Video_*Codec*

Netflix. (n.d.). Recomendações de velocidade de ligação à internet. Retrieved from

https://help.netflix.com/pt-pt/node/306

Ohm, J.-R., & Sullivan, G. J. (2013). High efficiency video coding: the next frontier in video compression [Standards in a Nutshell]. *IEEE Signal Processing Magazine*, *30*, 152–158. Retrieved from https://ieeexplore.ieee.org/iel5/79/6375903/06375943.pdf

Online, E. (2017). Portugueses já usam mais fibra ótica do que cabo. Retrieved from https://www.sapo.pt/noticias/economia/portugueses-ja-usam-mais-fibra-otica-do-que_5947faa6208166407768cc29

Paul Read, M.-P. M. (2000). *Restoration of Motion Picture Film*. (M.-P. Meyer, Ed.). Retrieved from https://books.google.pt/books?id=jzbUUL0xJAEC&pg=PA24

Pires, J. (2011). *Redes de Telecomunicações*. Retrieved from https://fenix.tecnico.ulisboa.pt/downloadFile/3779576319187/Cap4_RT_10.pdf

Practice, S. I. N. (2011). *The Handbook of MPEG Applications*. https://doi.org/10.1002/9780470974582

Público, C. C. S. (2017). Internet fixa em Portugal é das mais rápidas do mundo. A móvel, nem por isso. *Público*. Retrieved from https://www.publico.pt/2017/08/12/tecnologia/noticia/portugal-esta-entre-os-30-paises-com-acesso-fixo-mais-rapido-a-internet-nas-redes-moveis-nem-por-isso-1782185

Qualcomm. (2018a). No Title. Retrieved from https://www.qualcomm.com/snapdragon/devices/all

Qualcomm. (2018b). Snapdragon Platform Comparison. Retrieved from https://www.qualcomm.com/snapdragon/processors/comparison

Regula, O. (2016). Handbook on Digital Terrestrial Television Broadcasti ng Networks and Systems Implementati on Editi on of 2016.

Reilly, T. O. (2015). For Immediate Release.

Ribeiro, N. M; Torres, J. (2009). Tecnologias de Compressão Multimédia. ed. 1, 1 vol.,

ISBN: 978-972-722-633-7. Lisboa: FCA - Editora de Informática, Lda..

Schulzrinne, H. (2001). Internet Media-on-Demand : The Real-Time Streaming Protocol, 1–50.

Schulzrinne, H., Rao, A., & Lanphier, R. (1998). Real Time Streaming Protocol (RTSP), 1–92.

Specification, T. (2009). Specification for the use of Video and Audio Coding in Broadcasting Applications based on the MPEG-2 Transport Stream, *1*, 1–163.

Thomson, S. (n.d.). Czech public broadcaster begins transition to DVB-T2 and HEVC. Retrieved from https://www.digitaltveurope.com/2018/03/28/czech-public-broadcaster-begins-transition-to-dvb-t2-and-hevc/

Troncy, R., Huet, B., & Schenk, S. (2011). *Multimedia Semantics Metadata, Analysis and Interaction*. Wiley.

Union, I. T. (2003). H.264 (05/2003).

Vanam, R. (2007). *Motion Estimation and Intra Frame Prediction in H.264/AVC Encoder*. Retrieved from https://courses.cs.washington.edu/courses/csep590a/07au/lectures/rahullarge.pdf

W. Barz, H., & A. Bassett, G. (2016). *Multimedia Networks and Their Applications*. Wiley.

Waldrop, M. M. (2016). The chips are down for Moore's law. *Nature*. Retrieved from https://www.nature.com/news/the-chips-are-down-for-moore-s-law-1.19338

Wu, D., Member, S., Hou, Y. T., & Zhu, W. (2001). Streaming Video over the Internet : Approaches and Directions, *11*(3).

YouTube. (2018). AV1 Beta Launch Playlist. Retrieved from https://www.youtube.com/playlist?list=PLyqf6gJt7KuHBmeVzZteZUlNUQAVLwrZS