Universidade de Aveiro
2008
Departamento de Electrónica, Telecomunicações e Informática

TIAGO ANDRÉ GANDARINHO CAÇOILO

Sistema de Conversão de Vídeo

Universidade de Aveiro
2008
Departamento de Electrónica, Telecomunicações e Informática

TIAGO ANDRÉ GANDARINHO CAÇOILO

Sistema de Conversão de Vídeo

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Electrónica Telecomunicações. Trabalho realizado sob a orientação científica do Professor Dr. António José Nunes Navarro Rodrigues, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e Investigador do Instituto de Telecomunicações.

O Júri

Presidente

Prof. Dr. Rui Jorge Morais Tomaz Valadas
Professor Associado com Agregação na Universidade de Aveiro

Prof. Dr. António José Nunes Navarro Rodrigues
Professor Auxiliar da Universidade de Aveiro

Prof. Dr. Fernando José Pimentel Lopes
Professor Coordenador do Departamento de Engenharia Electrotécnica do Instituto Superior de Engenharia de Coimbra

Agradecimentos

Cabe-me aqui a minha sincera gratidão a todos aqueles que, de algum modo, colaboraram comigo ao longo deste trabalho.

Ao professor António Navarro, meu orientador no decorrer deste trabalho, pela oportunidade que me deu, pela disponibilidade apresentada e pelas condições que me proporcionou na realização deste trabalho. Agradeço, também, pelos conhecimentos que me foram sendo transmitidos ao longo da realização deste trabalho, bem como a revisão atenta que concedeu a esta dissertação.

Aos meus pais pelo apoio que me deram sempre no decorrer da minha formação, quer pessoal quer académica.

À minha irmã Andreia pela paciência.

À minha namorada Rute pelo apoio incondicional, prestado nos momentos bons e maus que surgiram no decorrer deste trabalho.

Aos meus colegas Luís Reis, Nuno Coelho e Nelson Cabral, pela ajuda de carácter técnico que concederam para o desenvolvimento deste trabalho científico.

Palavras-chave

Alta - Definição, CIF, H.264, HD, MPEG-4 AVC, RTP, SD, Transmissão de vídeo, Vídeo.

Resumo

Esta dissertação é composta por uma descrição dos conceitos base dos formatos de vídeo, uma explicação detalhada da norma MPEG-4 AVC/H.264, um estudo com algum detalhe do protocolo de transporte RTP e uma exposição rigorosa do Conversor de Vídeo implementado e respectivas conclusões.

O conversor de vídeo descrito nesta dissertação é um contributo para a implementação de um sistema de transcodificação de MPEG-4 AVC/H.264 escalável em MPEG-4 AVC/H.264 não escalável. Trata-se, contudo, de um sistema que selecciona os *streams* correspondentes a determinadas resoluções e qualidades de vídeo e os disponibiliza ao utilizador. Um exemplo típico é a difusão de televisão digital em formato H.264 escalável com duas camadas, a base em SD e a melhorada em HD.

A técnica de compressão de vídeo utilizada no presente trabalho foi a norma H.264 dado que possui uma elevada taxa de compressão.

Keywords CIF, Definition, H.264, HD, High- Video Transmission, MPEG-4 AVC, RTP, SD, Video.

Abstract

This dissertation is composed of a description of the video basic concepts, a detailed explanation of the standard MPEG-4 AVC/H.264, a detailed study for the transport protocol RTP and the video converter. Finally, it reports some conclusions that we have drawn.

The video converter proposed in this thesis is a contribution to the implementation of a transcoding system from scalable MPEG-4 AVC/H.264 into nonscalable MPEG-4 AVC/H.264. This converter selects streams with different qualities and spatial resolutions and provides them to the user. A typical example is in the digital television broadcasting environment where scalable H.264 with two layers are broadcasted using the base in SD and the enhancement in HD resolutions.

The video technique applied in the present work was the H.264 standard, because it provides a high video compression.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

Nomenclatura

2D	Duas Dimensões
APP	Application Defined
ASO	Arbitrary slice ordering
BYE	Membership Management
CABAC	Context-based Adaptive Binary Arithmetic Coding
CAE	Context-based Arithmetic Encoding
CAVLC	Context Adaptive Variable Length Coding
CCD	Charger Couple Device
CIE	Commission Internationale de L'Éclairage
CIF	Common Intermediate Format
CNAME	Nome canónico
CoDec	Encoder e Decoder
CRT	Cathode Ray Tube
CSRC	Contributing Source Identifiers
DCT	Discrete Cosine Transform
DPCM	CODEC de vídeo com compensação e estimação de movimento
DSCQS	Double Stimulus Continuous Quality Scale
DSL	Digital Subscriber Line
DV	Digital Video
DVD	Digital Versatile Disk
DWT	Discrete Wavelet Transform
FIR	Finite Impulse Response
FMO	Flexible Macroblock Order
FQ	Forward Quantiser
GB	Giga Byte
Gb	Giga bit
HD	High Definition video (Vídeo de Alta-definição)
HDTV	Televisão de Alta-Definição
HSL	Hue, Saturation and Lightness
HSV	Hue, Saturation and Value
IDCT	Inverse Discrete Cosine Transform
IDR	Instantaneous Decoding Refresh
IETF	Internet Engineering Task Force
INTRA	Codificação de *frames* de vídeo sem recurso a predição temporal
IP	Internet Protocol

IQ	Inverse Quantiser
ISDN	Integrated Services Digital Network
ISO	International Organization for Standardization
ITU	International Telecommunications Union
ITU-R	International Telecommunication Union Radiocommunication Sector
JPEG-LS	Joint Photographic Experts Group
LAN	Local Area Network
MPEG	Motion Video Expert Group
MV	Motion Vector (Vector de Movimento)
MVD	Motion Vector Difference
NAL	Network Abstraction Layer
NALU	Network Abstraction Layer Unit
NVP-II	Network Voice Protocol, versão 2
PAL	Phase Alternating Line
PES	Packetised Elementary Stream
PSNR	Peak Signal to Noise Ratio
QCIF	Quarter Common Intermediate Format
QOS	Quality of Service
QP	Passo de Quantificação
RGB	Red, Green and Blue colour space
RR	Receiver Report
RTCP	Real-Time Transport Control Protocol
RTP	Real Time Protocol
RTP/IP	Internet Real/Time Transport Protocol
RVLC	Códigos de Comprimento Variável Reversíveis
SD	Standard Definition video
SDES	Source Description
SDP	Session Description Protocol
SP/SI	Synchronization/Switching Pictures
SQCIF	Sub Quarter Common Intermediate Format
SR	Sender Report
SSRC	Synchronization Source Identifier
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VCEG	Video Coding Experts Group
VCL	Video Coding Layer
VLC	Codificador de comprimento variável
YC_bC_r	Luminância, Crominância Azul e Crominância Vermelha

Capitulo 1 – Introdução

O desenvolvimento de métodos de codificação de vídeo têm vindo ao longo dos tempos a desempenhar uma importante função ao nível da evolução dos sistemas de transmissão e recepção de vídeo. Este facto tornou possível a existência de sistemas de transmissão de vídeo com co-existência de outros serviços, como é o caso da internet.

Com o conhecimento de codificação e transmissão, torna-se evidente a noção de quanto maior a taxa de codificação maiores são os recursos disponíveis para outros serviços, aumentando a rentabilidade da rede do operador. Por conseguinte, o grupo MPEG (*Motion Video Expert Group*) tem vindo a incentivar o desenvolvimento das normas MPEG-4 AVC/H.264 e a MPEG-4 Visual. Estas permitem taxas de compressão superiores às que actualmente se encontram em uso na indústria (MPEG-2). Contudo estas normas necessitam de elevados recursos computacionais, tanto para a codificação como para a transmissão e recepção. A dificuldade criada pelas normas obrigou o desenvolvimento em simultâneo de protocolos de rede que tendem a atenuar as perdas temporais criadas pela compressão, como é o caso do protocolo de transporte RTP (*Real Time Protocol*). Este permite a transmissão de elevadas quantidades de informação com utilização de poucos recursos computacionais, permitindo ainda o tratamento de cada pacote RTP de forma simples e eficaz.

O trabalho aqui apresentado tem, desta forma, como objectivo o desenvolvimento de uma aplicação de software para tratamento de serviços de vídeo transmitidos via utilização do protocolo RTP. Esta aplicação tem o intuito de reduzir os tempos de espera no arranque dos serviços de vídeo em plataformas de baixos recursos de hardware, adicionar alguma robustez e segurança no acesso a serviços não contratados pelos clientes, isto é, se o cliente apenas receber os conteúdos a que tem direito pela assinatura do serviço não existe a possibilidade de utilização não autorizada de outros serviços de vídeo existentes na rede.

Assim, para melhor compreensão do trabalho desenvolvido, esta dissertação começa por descrever os conceitos base de vídeo, como é o caso da amostragem, o espaço de cor, os formatos de vídeo e a qualidade de vídeo. De seguida é apresentado e explicado o princípio da codificação de vídeo e o respectivo codificador, bem como os módulos necessário à compatibilidade com sistemas anteriores às normas mais recentes (o modelo baseado em macroblocos – DPCM/DCT). Após a assimilação dos conceitos anteriores, passa-se à apresentação de um estudo detalhado da norma MPEG-4 AVC/H.264, seus mecanismos de transporte e armazenamento de conteúdos. Com isto, encontra-se preparado para encarar o protocolo de transporte RTP e todos os seus detalhes.

Terminado o estudo dos conceitos apresentados anteriormente para o desenvolvimento do Conversor de vídeo (Conversor de Formatos de Vídeo), apresenta-se a plataforma de hardware e respectivo software utilizados como base de desenvolvimento deste projecto. Por fim expõe-se o Conversor de formatos criado, o seu conceito,

algoritmo, implementação realizada, código desenvolvido, demonstradores de funcionamento e respectivos testes de desempenho ao sistema.

Esta dissertação termina com as conclusões retiradas da realização deste trabalho e respectivo alcance dos objectivos propostos.

Capitulo 2 – Sinais de Vídeo - Conceitos Básicos

Neste capítulo apresentam-se todos os princípios básicos necessários para a realização de uma boa compressão e codificação de um sinal de vídeo.

Inicialmente descreve-se o princípio da amostragem em sinais de vídeo e uma descrição do que significa espaço de cor (*colour space*). Apresenta-se também uma descrição dos diversos formatos de vídeo e suas implicações na codificação do sinal de vídeo. Por último é apresentada uma análise comparativa da qualidade do vídeo antes e depois de uma codificação.

2.0 – Corpo Humano – A Visão

No corpo humano, o olho (Figura 1) funciona como um sensor óptico e o cérebro como um processador. No olho humano, a luz passa pela íris, é focada pelo cristalino e projecta-se na retina. Na retina existem dois tipos de "sensores", células sensíveis à luz: os bastonetes (*rods*) e os cones. Os bastonetes permitem a visão escotópica, e os cones que permitem a visão fotópica. Estes sensores, existem em quantidades diferentes e apresentam diferentes distribuições na retina. No olho existe apenas um tipo de bastonetes, enquanto que cones existem três tipos diferentes, vermelhos (≈64%), verdes (≈33%) e Azuis (≈2%) do olho [1][2][3].

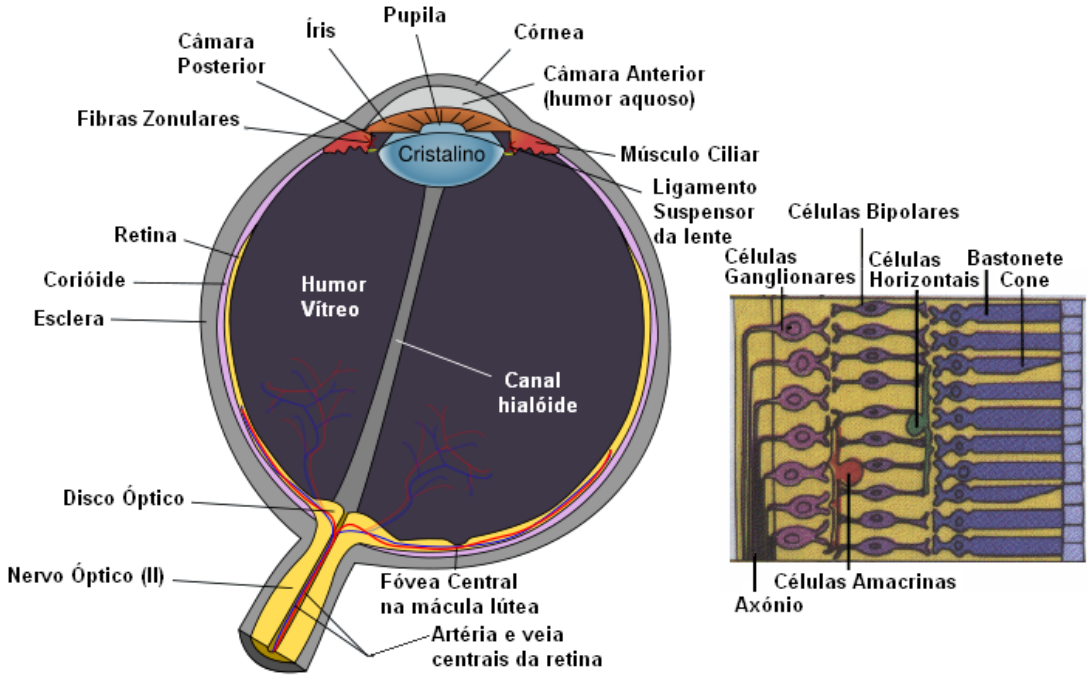

Figura 1 - Descrição do Olho Humano (adaptada de [3])

Os cones e os bastonetes são fotorreceptores que libertam moléculas neurotransmissoras a uma taxa máxima na escuridão, diminuindo com o aumento da luminosidade.

Do conhecimento anatómico, sabe-se ainda que olho humano é sensível à radiação electromagnética numa pequena gama de espectro, gama do visível (dos 400nm aos 700nm) (Figura 2), muitos animais vêm noutras partes do espectro electromagnético, como é o caso das abelhas que são sensíveis aos Ultra Violeta (do 1nm aos 400nm) e os cães e gatos que são pouco sensíveis à cor.

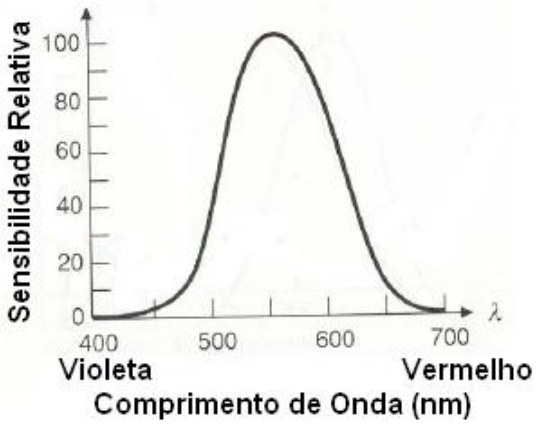

Figura 2 - Eficiência Luminosa do Olho humano [1]

Assim o processamento de sinais de vídeo tira partido de todas as características do olho humano de forma a conseguir um melhor desempenho visual com a menor quantidade de recursos.

2.1 - Amostragem

Um sinal de vídeo consiste numa sequência de imagens reais obtidas por um processo de amostragem espacial e temporal. A amostragem espacial é o registo de imagens ou fatias (*slices*), enquanto que a amostragem temporal, se baseia no número de imagens capturadas por segundo que ao serem reproduzidas criam uma sequência, obtendo-se a ilusão de movimento (Figura 3).

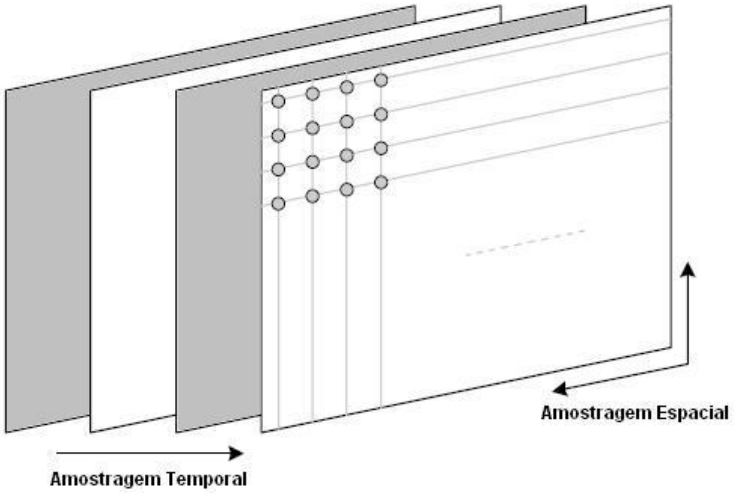

Figura 3- Amostragem Temporal e Espacial de uma sequência de vídeo (adaptada de [5])

A cada amostra espacial (pontos a cinzento na Figura 3) chamamos de pixel, da língua inglesa *picture element*, a um conjunto de píxeis inseridos no mesmo plano espacial chamamos de amostra temporal. Quando a transmissão de vídeo só se dava numa escala de cinzentos, cada pixel representava uma componente cujo valor significava a quantidade de brilho. Actualmente, a maior parte dos sistemas de vídeo funciona a cores, onde cada pixel representa três componentes, das quais os seus valores se encontram relacionados com a percepção do olho humano. O valor da componente de um pixel depende do brilho e da cor da região onde a amostra é realizada [4].

Do número de amostras espaciais retira-se o que se chama de qualidade da imagem, enquanto que das amostras temporais absorve-se a qualidade da fluidez de movimentos, isto é, uma cena amostrada espacialmente com poucas amostras apresenta um aspecto mais rugoso (Figura 4) do que uma imagem com um maior número de amostras espaciais (Figura 5) [5]. Se a amostragem temporal de uma cena for realizada com poucas amostras por segundo, por exemplo 10 ou 15 (taxa de amostragem típica de comunicações de baixo débito), verifica-se a presença de "saltos" entre movimentos. Esta situação não se verifica quando se realiza uma amostragem com 25 a 30 amostras por segundo (frequência normalmente usada nas transmissões de televisão actuais). Este fenómeno deve-se à taxa de actualização de imagens do olho humano, isto é, quando duas imagens são projectadas sucessivamente e num curto período de tempo o olho humano retém a imagem precedente durante alguns instantes, cerca de 1/10 de segundo. Esta retenção provoca na pessoa que se encontra a visionar o vídeo uma sensação de continuidade entre frames [6].

Actualmente, com a televisão de alta definição, utiliza-se amostragem temporal com 50 a 60 frames por segundo que associadas às características do olho humano, anteriormente descritas, aumentam a sensação de movimento fluído, suave e sem "saltos" [5].

Figura 4 – Amostragem Espacial com baixo número de amostras por frame (adaptada de [5])

Figura 5 – Amostragem Espacial com 25 frames por segundo (adaptada de [5])

2.1.1 – Frames e Campos

O aumento do número de imagens por segundo aumenta consequentemente a carga de dados a processar, implicando desta forma, necessidades de melhores e mais dispendiosos recursos. Assim, de maneira a minimizar este problema, existe uma técnica onde a imagem não é amostrada de uma forma progressiva (amostragem progressiva implica amostrar cada frame totalmente) mas sim por campos, do inglês *fields*. Este tipo de amostragem tem o nome de amostragem entrelaçada. O método de amostragem entrelaçada obriga a que durante cada intervalo de amostragem temporal, apenas um campo seja amostrado, sendo o próximo campo amostrado no intervalo seguinte. Um exemplo de campo é a divisão entre linhas pares e linhas ímpares de uma frame de vídeo completa (Figura 6). Uma vez que cada campo contém apenas metade da informação, consequentemente conterá metade da carga necessária para transmissão, para igual número de imagens por segundo.

A vantagem na utilização deste método é a possibilidade de enviar o dobro da informação por segundo em relação ao número de frames em modo progressivo (para uma taxa de transferência igual), ficando-se assim com uma sensação de um movimento mais suave quando nos encontramos a visualizar a sequência de vídeo. Exemplo desta técnica é o sistema PAL (*Phase Alternating Line*), utilizado em Portugal na transmissão de televisão analógica. Este sistema utiliza 50 campos por segundo, implicando que quando se está a visualizar uma sequência de vídeo esta pareça muito mais suave nas transições do que a mesma sequência a ser transmitida com amostragem em modo progressivo, a uma taxa de 25 frames por segundo [5].

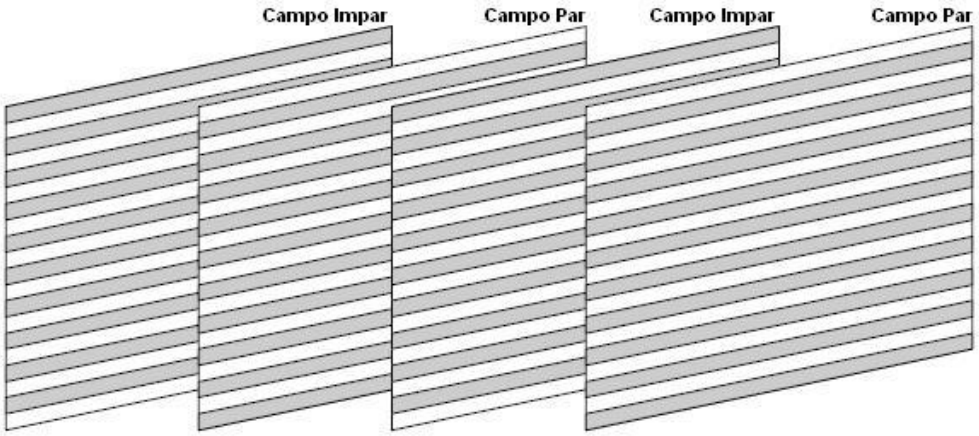

Figura 6 – Sequência de vídeo entrelaçada (adaptada de [5])

2.2 – Espaço de Cor (*Colour Space*)

O vídeo, tal como o conhecemos actualmente, é um composto de cor e brilho, para isso foi necessário criar mecanismos de captura e de representação da informação adquirida.

Em aplicações onde se pretende apenas vídeo monocromático, cada pixel contém apenas a informação relativa ao brilho, ou seja, valores de pixel mais elevados representam pontos com maior intensidade luminosa e valores mais baixos pontos de menor intensidade.

Com o aparecimento do sistema a cores foi necessário representar a cor. À forma de representar a cor e o brilho (luminância ou *luma*) chamou-se de espaço de cor, ou do inglês *Colour Space*. O espaço de cor consiste na representação matemática das cores tal como as conhecemos. Com o desenvolvimento das diversas tecnologias ao longo dos anos foram sendo criados diferentes espaços de cor, cada um apropriado a uma determinada aplicação. Neste capítulo serão apresentados os mais utilizados actualmente.

2.2.1 – RGB (*Red, Green and Blue*)

No espaço de cor RGB, a cor da amostra é representada por três números que identificam nas devidas proporções o vermelho, o verde e o azul (as três componentes primárias da luz) (Figura 7).

Figura 7 - Componentes de cor (vermelho, verde e azul) do espaço de cor RGB (adaptada de [7])

A figura anterior mostra a componente vermelha (***R**ed*), verde (***G**reen*) e azul (***B**lue*) da imagem principal. A componente vermelha representa todas as amostras vermelhas presentes na imagem, a componente verde contém todas as amostras verdes e a azul as respectivas amostras azuis. Se por algum motivo fossem adicionadas as três componentes em simultâneo, obter-se-ia a imagem real (paisagem). Este espaço de cor pode ser ainda representado por um sistema de coordenadas cartesiano tridimensional, onde cada eixo corresponde a uma das três cores aditivas. Quantidades diferentes de cada componente representam uma cor diferente. Ao ser definido um vector neste espaço de cor, que tenha igual quantidade de cada componente, o vector representa a variação de intensidades de brilho, ou seja, escalas ou tonalidades de cinzento (Figura 8).

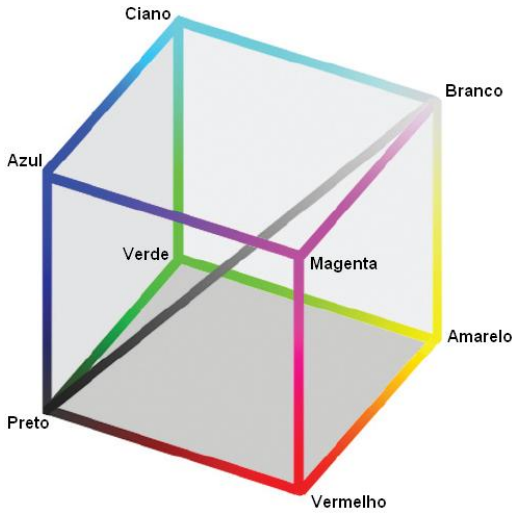

Figura 8 – RGB - Cubo da Composição das Cores (adaptada de [8])

Cada pixel da imagem, usando o RGB, é constituído por três valores (vermelho, verde e azul). Este tipo de representação é o ideal na utilização de sistemas de captura e apresentação de imagens compostas de diversas cores, uma vez que os ecrãs utilizam RGB para representar imagens e as câmaras de captura usam CCDs (*Charger Couple Devices)* com sensores dedicados de R, G e B para a aquisição.

A utilização do RGB implica ainda que cada componente (vermelha, verde e azul) possua a mesma largura de banda, ainda que a imagem não possua numa das componentes muita informação. Esta situação provoca mau desempenho do RGB na representação de imagens reais (naturais), o que implica outro problema, se o utilizador desejar realizar um ajuste de brilho do vídeo, torna-se necessário proceder ao ajuste das três componentes, atrasando assim todo o processamento da imagem.

Devido ao mau desempenho do RGB para imagens reais, tornou-se necessário desenvolver uma nova técnica que colmatasse o mau funcionamento do espaço de cor, assim apareceu o espaço de cor YC_bC_r [4][5].

2.2.2 – YCbCr

O olho humano, é mais sensível à luminância (brilho) do que propriamente à cor. Do RGB sabe-se que as cores são repartidas de igual forma, o que implica que são normalmente armazenadas com a mesma resolução. Actualmente sabe-se que é mais eficiente separar as componentes de cor do brilho e dar mais valor ao mesmo em detrimento das componentes de cor.

O espaço de cor YC_bC_r, por vezes referido como YUV, e suas variações, tornou-se popular por beneficiar a luminância em detrimento da cor, representando assim de uma forma eficiente as imagens a cores. *Y* representa a componente luminância (*luma*). Esta componente pode ser calculada pela média ponderada de R (vermelho), G (verde) e B (azul), segundo a expressão:

$$Y = k_r R + k_g G + k_b B \tag{2.1}$$

Onde os factores *k* da equação 2.1 correspondem aos pesos de cada componente de cor vermelha, verde e azul.

A informação de cor pode ser ainda representada como a diferença entre cores, ou seja, pela sua crominância, também conhecida como *chroma*, do inglês *chrominance*. Cada factor de crominância é a diferença entre as componentes R, G, B e a luminância, como mostram as expressões seguintes:

$$\begin{aligned} C_b &= B - Y \\ C_r &= R - Y \\ C_g &= G - Y \end{aligned} \tag{2.2}$$

A descrição completa de uma imagem a cor é obtida pelo conjunto de informações de *Y* (componente de luminância) e pelas três componentes de cor C_b (crominância azul), C_r (crominância vermelha) e C_g (crominância verde). Este conjunto de informação representa a diferença entre a intensidade de cor e a média da luminância em cada amostra da imagem.

Com a utilização do espaço de cor YC_bC_r, verifica-se o aparecimento de mais uma componente comparativamente com o espaço de cor RGB. Apesar desta situação se verificar o espaço YC_bC_r é mais eficiente. Devido à existência da constante $C_b+C_r+C_g$, apenas duas das componentes são necessárias armazenar, ou transmitir (C_b e C_r) para além da luminância (*Y*). A terceira componente pode ser calculada à posterior a partir das outras duas.

No espaço de cor YC_bC_r, apenas a luminância (*Y*) e as componentes de crominância azul e vermelha (C_b e C_r) são transmitidas. YC_bC_r apresenta uma grande vantagem em relação ao RGB, isto porque as componentes de crominância são representadas com menor resolução, restando assim uma maior resolução para que a componente de luminância seja transmitida. Sendo a condição ideal para o olho humano, visto ser mais sensível a impulsos de luminância do que a impulsos de crominância. Este facto permite criar uma compressão

de dados, que em comparação com o espaço de cor RGB, não afecta o vídeo de forma considerável, isto é, um utilizador casual não detecta qualquer diferença, apenas utilizadores mais experientes detectam diferenças ténues em relação ao RGB.

O YC_bC_r, como se tem vindo a mencionar, está directamente relacionado com o RGB (equações 2.3 e 2.4). Para tal existem relações matemáticas que demonstram a relação de cada componente de YC_bC_r com as de RGB.

Visto que, $k_b + k_r + k_g = 1$ e partindo da equação 2.1, tem-se:

$$Y = k_r R + (1 - k_b - k_r)G + k_b B$$

$$C_b = \frac{0.5}{1-k_b}(B - Y) \tag{2.3}$$

$$C_r = \frac{0.5}{1 - k_r}(R - Y)$$

$$R = Y + \frac{1 - k_r}{0.5} C_r$$

$$G = Y - \frac{2k_b(1-k_b)}{1-k_b-k_r} C_b - \frac{2k_r(1-k_r)}{1-k_b-k_r} C_r \tag{2.4}$$

$$B = Y + \frac{1 - k_b}{0.5} C_b$$

Da recomendação do ITU-R BT.601 [9], sabe-se que k_b=0.114 e k_r=0.299, substituindo este valores nas equações 2.3 e 2.4 obtém-se:

$$Y = 0.299R + 0.587G + 0.114B$$

$$C_b = 0.564(B - Y) \tag{2.5}$$

$$C_r = 0.713(R - Y)$$

$$R = Y + 1.402C_r$$

$$G = Y - 0.344C_b - 0.714C_r \tag{2.6}$$

$$B = Y + 1.772C_b$$

O cálculo destes factores pode ser analisado de uma forma visual na Figura 9. Nesta figura pode observar-se a componente da luminância (B) obtida de (A), verificando-se que os pontos mais brilhantes de (A) são apresentados com cinzentos mais claros e os pontos mais escuros com cinzentos mais escuros. Em (C) e para a crominância azul, C_b, verifica-se que os cinzentos mais claros correspondem a zonas de maior intensidade de azul, enquanto que para a crominância vermelha (D) os pontos de maior intensidade de cor vermelha são representados com zonas a cinzento mais claro [5].

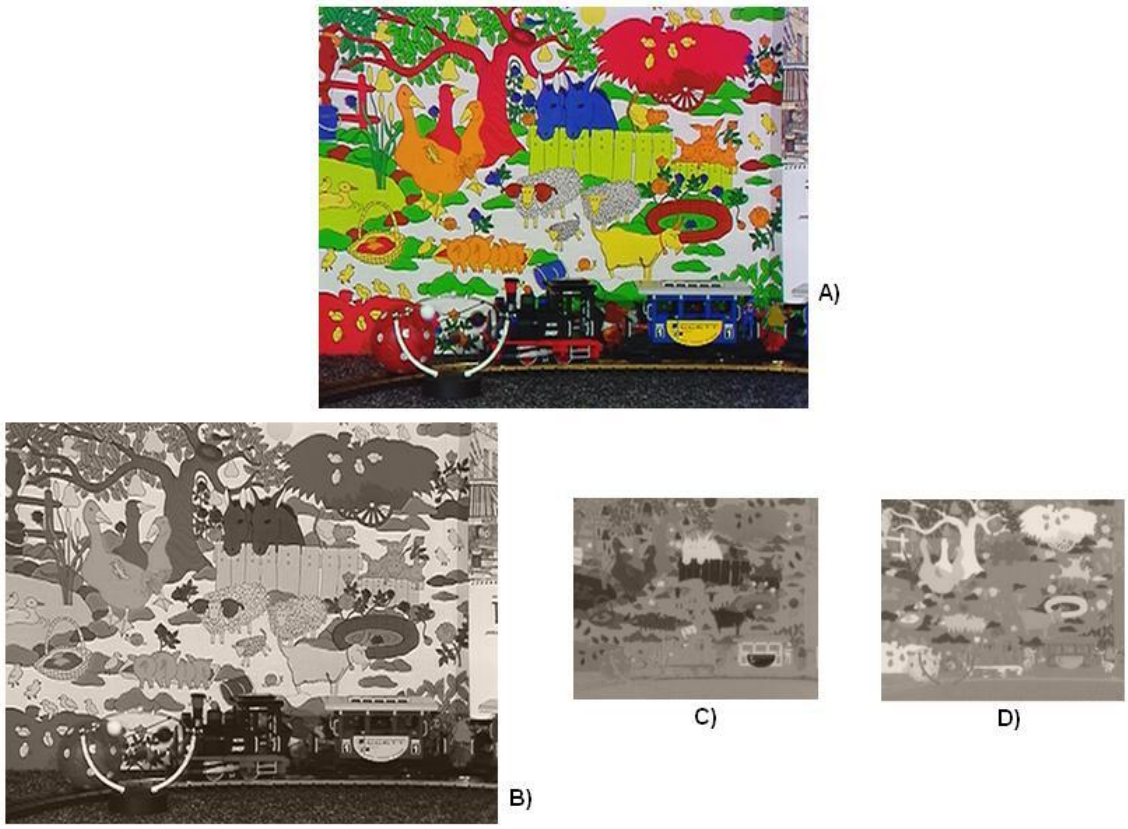

Figura 9 – Decomposição de imagem a cores (A) em componentes: luminância - *Y* (B), crominância azul - C_b (C) e crominância vermelha - C_r (D)

2.2.3 – HSL e HSV

HSL e HSV derivam do espaço de cor RGB. Estes espaços de cor encontram-se directamente relacionados, pelo que serão explicados em simultâneo. O HSL e o HSV são uma forma de representar a imagem de uma forma perceptual, a cor RGB.

O HSL caracteriza a imagem em termos de matiz (tonalidade), saturação e brilho (*L*, luminância), em contrapartida o HSV caracteriza a imagem pela sua tonalidade, saturação e pelo seu valor, isto é, o factor no qual a cor coincide ao longo do eixo claro-escuro (brilho).

HSV e HSL descrevem cores como pontos de um cilindro cujo eixo central altera de preto na base inferior ao branco no topo, com cores neutras entre eles. O ângulo em torno do eixo corresponde à tonalidade (*H*, do inglês *hue*), a distância entre o eixo corresponde à saturação (*S*) e distância ao longo do eixo corresponde ao valor (*V*) (Figura 10) [8][10].

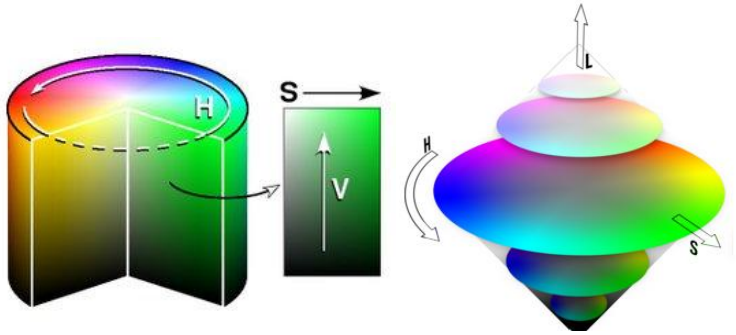

Figura 10 – Caracterização do HSV (imagem à esquerda) e do HSL (imagem da direita) (adaptada de [11])

O HSV ou o HSL podem ser muito úteis quando se torna necessário remover certas características de uma imagem, que não podem ser alcançadas na utilização de um espaço de cor como o RGB, devido à sua gama de cor por pixel diversificada.

Um exemplo de HSV e HSL pode ser visualizado no Anexo 1.

2.2.4 – CIE (*Commission Internationale de L'Éclairage*)

CIE é um espaço de cor, dos primeiros a ser criado e utilizado na indústria. O CIE é um diagrama cromático, representado segundo um sistema de coordenadas cartesiano (xyz) ao qual corresponde o campo da cor com um espaço bidimensional (eixos xy). Como se pode ver na Figura 11, o campo luminosidade aparece representado pelo terceiro eixo (z).

Em comparação com o sistema YUV, a componente luminância apresenta-se no CIE como uma imagem monocromática em que os pontos mais claros representam campos de maior brilho. Enquanto que pontos mais escuros representam campos de menor intensidade luminosa. Em contrapartida as duas componentes primárias, x e y, no espaço de cor CIE, admitem somente valores positivos, em detrimento das componentes U e V do YUV. Estes valores podem ser combinados, formando qualquer cor do espectro electromagnético.

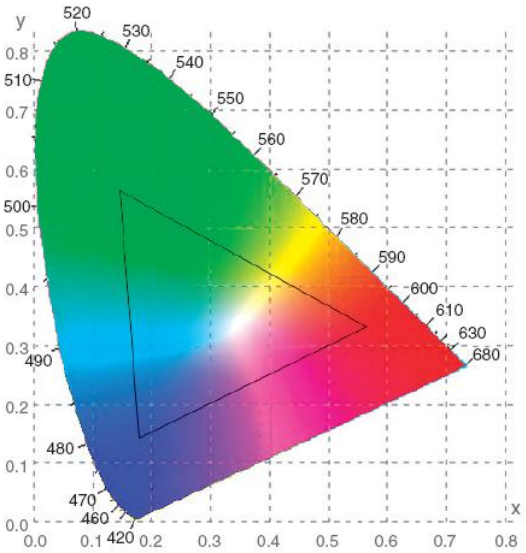

Figura 11 - Diagrama Cromático do Espaço de cor – CIE (adaptada de [8])

Na figura anterior pode-se verificar que as cores se encontram totalmente saturadas ao longo da borda da representação multicolor. Os números representam o comprimento da onda de luz do espectro electromagnético, em nanómetros (a cor). Pode-se ainda observar na figura, um triângulo inscrito que representa as cores típicas de representação de um monitor do tipo CRT (tubo de raios catódicos, do inglês *Cathode Ray Tube*). Os vértices do triângulo correspondem às máximas emissões que o fósforo do tubo de raios catódicos consegue transmitir.

O CIE é muito utilizado em sistemas de instrumentação. Estes sistemas utilizam o CIE para definir a onda dominante e a sua pureza, servindo este espaço de cor (CIE) para medir, por exemplo, a máxima resolução de cor que um monitor consegue transmitir.

2.3 – YC_bC_r - Formas de Amostrar um Sinal de Vídeo

Numa imagem em RGB, as diferentes componentes apresentam a mesma resolução, o que implica que por cada amostra de uma imagem existam três componentes de igual tamanho, representando as componentes RGB da imagem real. No entanto, o espaço de cor YC_bC_r permite ter resoluções diferentes para cada componente de luminância e crominância. Em YC_bC_r existe uma nomenclatura, $Y:C_b:C_r$, que permite a identificação rápida de cada formato. Por exemplo se o utilizador tiver um formato de amostragem de 4:4:4, implica que por cada posição de pixel existe uma amostra de cada componente, Y C_b e C_r (Figura12).

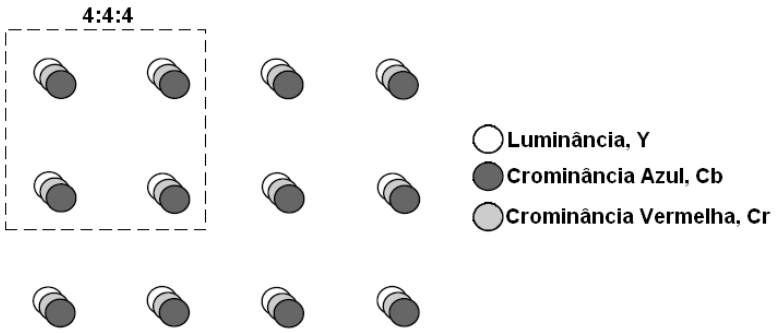

Figura 12 - Formato 4:4:4 (adaptada de [5])

A forma de representação, $Y:C_b:C_r$ tornou-se um standard entre a indústria, de tal forma que ainda é aplicada nas técnicas mais recentes de codificação de vídeo como é o caso do MPEG-4 Visual e o MPEG-4 AVC/H.264.

Para além do formato de amostragem 4:4:4, mencionado anteriormente, existem outras formas de amostrar como indicado na Tabela 1.

Tabela 1 - Formatos de Amostragem YCbCr (adaptada de [13])

Formato	Número de Píxeis de Y	Número de Píxeis de C_r ou C_b na vertical	Número de Píxeis de C_r e C_b na horizontal
4:4:4	4	4	4
4:2:2	4	4	2
4:2:0	4	2	2
4:1:1	4	4	1
4:4:4:4	4	4	4

O formato 4:4:4:4, apesar de apresentar um aspecto diferente do normal, possui as mesmas resoluções que o 4:4:4 com o acréscimo de uma componente K que serve para transmitir dados (*keying*).

O formato 4:2:2, conhecido também como YUY2, indica que as componentes de crominância têm a mesma resolução vertical que a luminância, mas metade da resolução horizontal, assim sendo 4:2:2 diz que para cada quatro amostras de luminância na direcção horizontal existem duas amostras de crominância azul (C_b) e duas amostras de crominância vermelha (C_r) (Figura 13). Normalmente este tipo de formato é usado em sistemas de reprodução de vídeo a cores de alta definição.

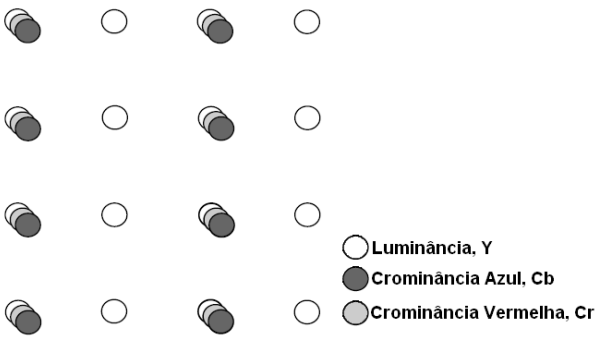

Figura 13- Formato 4:2:2 (adaptada de [5])

O formato 4:2:0, também conhecido como YV12, é um dos mais populares entre os diversos formatos de amostragem. Este formato apresenta uma C_b e C_r com metade da resolução horizontal e vertical de Y (Figura 14). Representar nesta forma, pode parecer um pouco confuso dado que a indicação de zero na posição de C_r poderia indicar que a mesma não se encontrava presente, este facto não se verifica, a componente de crominância vermelha existe. Esta forma de representar foi escolhida como um "código" para representar este caso particular de amostragem.

É comum verificar-se amostragem em formato 4:2:0 em aplicações de vídeo-conferência, televisão digital e em *Digital Versatile Disk* (DVD). Uma espécie de 4:2:0 é usada actualmente na norma MPEG-4 Visual e MPEG-4 AVC/H.264, para além de ser usado entrelaçamento da sequência de vídeo, as amostras de Y, C_b e C_r, pertencentes a uma frame de uma sequência de vídeo são alocadas em dois campos (Anexo 2).

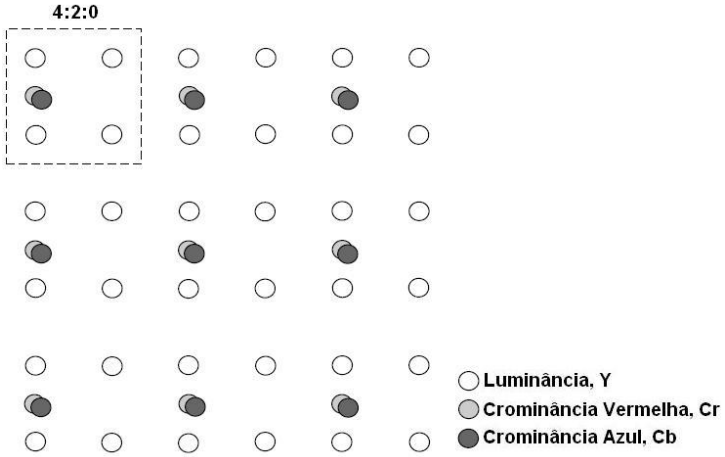

Figura 14 - Formato 4:2:0 (adaptada de [5])

O formato 4:1:1,conhecido como YUV12, é muito usado em aplicações de compressão de vídeo digital (DV – *Digital Video*) e de vídeo de consumo. Este formato indica que para cada quatro píxeis de luminância horizontal existe um pixel de C_b e de C_r mantendo a mesma resolução vertical (Figura 15) [5].

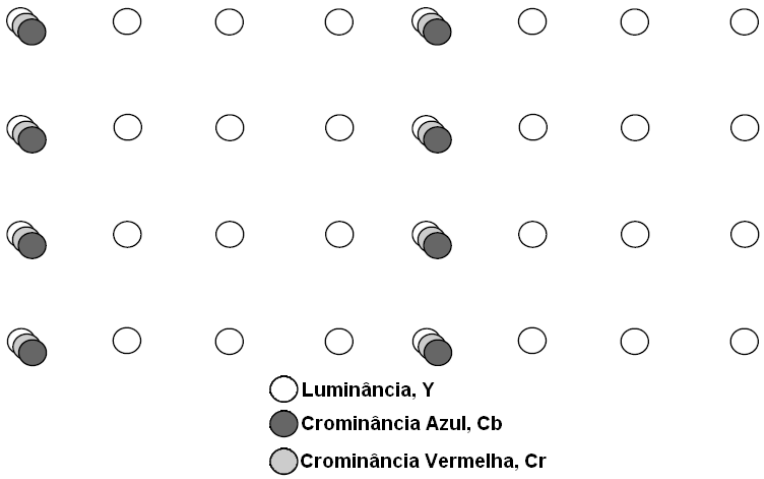

Figura 15 – Formato 4:1:1

2.4 – Formatos de Vídeo

Existe uma grande variedade de formatos de codificação de vídeo, mas dentro do contexto deste trabalho apenas serão abordados os formatos pertencentes ao grupo *Common Intermediate Format* (CIF).

Os formatos CIF são importantes uma vez que representam a base de uma série de formatos (Tabela 2).

Tabela 2 - Formato de Frames de Vídeo (adaptada de [5])

Formato	Luminância Resolução (horizontal x vertical)	Bits por frame (4:2:0, 8 bits por amostra)
Sub-QCIF	128 x 96	147456
Quarter CIF (QCIF)	176 x 144	304128
CIF	352 x 288	1216512
4CIF	704 x 576	4866048

A Figura 16 representa uma frame de luminância codificada nos diversos formatos CIF. A escolha de um tipo de resolução depende da aplicação, da capacidade de armazenamento disponível e da capacidade de transmissão. Por exemplo o 4CIF é apropriado para a transmissão de televisão e para o formato de vídeo – DVD. CIF e QCIF são populares para aplicações vídeo-conferência. QCIF e SQCIF são adequadas para aplicações de multimédia móvel onde a resolução do *display* e o *bitrate* são reduzidos. A Tabela 2 representa ainda o número de bits necessário para representar uma frame de vídeo descomprimida em cada formato (assumindo uma amostragem do tipo 4:2:0 com 8 bits de luminância e crominância). Por exemplo, para o formato Sub-QCIF o número de bits é igual a $\left((128 \times 96) \times 8 + 2 \times \frac{(128 \times 96)}{4} \times 8\right) = 147456$.

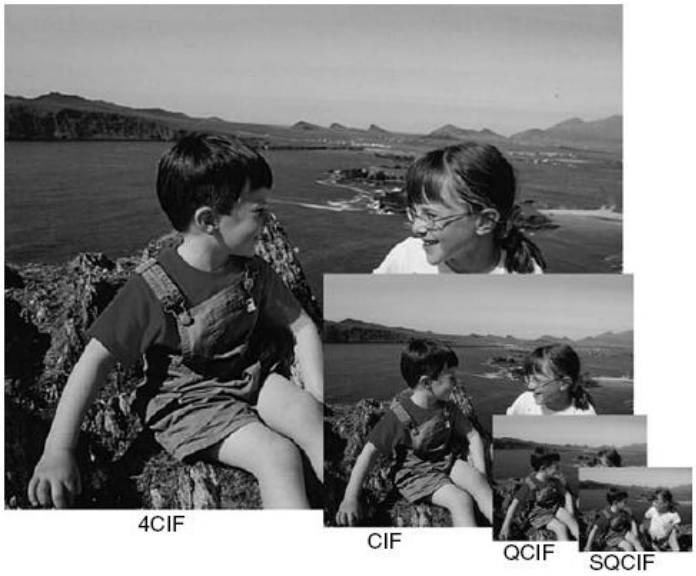

Figura 16 - Frame de Vídeo codificada em diversos formatos (adaptada de [5])

2.5 – Qualidade de Vídeo

A necessidade de caracterizar um sistema de vídeo, levou à criação da noção de qualidade de um sinal de vídeo. Esta noção não é fácil de aplicar, uma vez que a qualidade é classificada segundo um carácter subjectivo, isto é, um sinal de vídeo pode ser caracterizado por um observador como de elevada qualidade e por outro como sendo um sinal de baixa qualidade.

Devido a esta análise subjectiva, o ITU-R emitiu uma recomendação – ITU-R BT.500-11 [14] que protela o método de testes ao observador quando se encontra a analisar a qualidade de um sinal de vídeo, este método tem o nome de *Double Stimulus Continuous Quality Scale* (DSCQS), este método sugere que o observador seja sujeito a várias sequências de imagens do vídeo A e B. As imagens de vídeo são exibidas de forma aleatória sem identificação por parte do observador a que corresponde cada imagem, se ao vídeo A ou B. Com este método é ainda pedido ao observador que classifique a sequência dentro de uma escala sequêncial. No final do teste, realizado ao observador, são normalizados os resultados obtidos da análise por parte do mesmo, após este trabalho é atribuída uma classificação à sequência de vídeo. Este tipo de método, apesar de ter resultados satisfatórios, apresenta ainda assim alguns problemas, dado que se o observador for uma pessoa com alguma experiência na análise de sequências de vídeo irá realizar uma análise completamente diferente tentando sempre detectar os problemas do codificador, obtendo-se assim um resultado não isento. Estes tipos de situações podem ser sempre minimizados com a realização de um elevado número de testes a observadores diferentes. Estas análises impõem custos elevados e muito tempo dispendido em análise de resultados.

A qualidade de vídeo pode ser também avaliada de forma objectiva, utilizando-se para tal algoritmos de análise.

Um desses algoritmos é a análise da relação de pico do sinal/ruído, do inglês *Peak Signal to Noise Ratio* (PSNR),

$$PSNR_{dB} = 10log_{10}\frac{(2^n-1)^2}{MSE} \tag{2.7}$$

onde

$$MSE = \frac{\sum[f(i,j)-F(i,j)]^2}{N^2} \tag{2.8}$$

O PSNR é medido numa escala logarítmica e depende do erro médio quadrático (MSE) da imagem original e da frame manipulada, relativamente ao valor mais alto de sinal na imagem, elevado ao quadrado; *n* corresponde ao número de bits por amostra da imagem original (factor $(2^n\text{-}1)^2$).

Como se pode verificar o PSNR é fácil e rápido de se calcular, visto isto, o PSNR tornou-se um método muito popular usado para a comparação de imagens de vídeo comprimidas e descomprimidas.

Como cada método apresenta contrapartidas, o PSNR também as apresenta, nem sempre os valores obtidos pelo PSNR são coincidentes com os obtidos de forma subjectiva. Em alguns casos, onde a qualidade de imagem atribuída pelo PSNR é boa o efeito visual que se encontra na imagem não é o ideal, por exemplo, se na imagem se encontrar um rosto de uma pessoa com alguma distorção, para o observador já é um efeito negativo na imagem, classificando-a como de baixa qualidade. Isto porque o ser humano dá valor à perfeição do rosto em detrimento do resto (Anexo 3).

Para além do PSNR, existem outros métodos objectivos para análise da qualidade da imagem de vídeo, mas nenhum deles tem um desempenho superior ao PSNR, pelo que não serão mencionados neste trabalho [15].

Capitulo 3 – Codificação de Vídeo

Neste capítulo caracteriza-se a Codificação de Vídeo, parte-se da sua definição genérica e vai-se descrevendo passo a passo todo o conceito do que é a codificação de um sinal de vídeo, partindo do Modelo Temporal, passando pelo Modelo Espacial e pelo Codificador de Entropia.

Caracteriza-se ainda o modelo utilizado em codificadores de vídeo mais antigos e o seu impacto nos actuais.

3.1 – Compressão de Um Sinal de Vídeo – Definição

Dá-se o nome de compressão ou codificação de vídeo ao processo de compactação de um sinal de vídeo com um determinado número de bits num outro sinal com um menor número de bits, ou seja, é um processo de compactar ou condensar uma sequência digital de vídeo numa outra sequência com inferior número de bits. A compactação de uma sequência de vídeo digital em relação à não compactação desse mesmo vídeo apresenta vantagens, pois a largura de banda para transmissão de dados tem os seus custos (quanto maior a largura de banda mais caro se torna o processo de transmissão), como também a capacidade de armazenamento é limitada (apesar de actualmente o preço por GB ser reduzido, o número de filmes e sequências de vídeo no mercado tem aumentado exponencialmente).

O acto de comprimir envolve sempre um par de sistemas, o compressor ou codificador (*encoder*) e o descompressor ou descodificador (*decoder*).

O codificador converte os sinais originais numa sequência de dados comprimida (ocupando um número reduzido de bits), de forma poder-se transmitir a sequência de vídeo. Após a recepção da sequência de vídeo no descodificador dá-se a sua descompressão para a sua forma original.

Codificar e descodificar é uma tarefa importante, sendo cada vez mais necessário. O cliente deixou de querer esperar tanto tempo por um vídeo, ou seja, pretende poder visualizar determinado vídeo no instante imediato em que "carrega no botão" (redução do tempo de lançamento de serviço). Facto impulsionador da indústria para o desenvolvimento de algoritmos novos mais rápidos, cada vez mais eficazes na compressão e descompressão, assim como na diminuição de perdas no referido processo.

A compressão das sequências de vídeo implica menores quantidades de bits a transmitir, consequentemente maior velocidade de transmissão de dados.

Ao conjunto codificador e descodificador, dá-se o nome de *CODEC* (do inglês *enCOder/ DECoder*) (Figura 17).

Figura 17 - Sistema com Codificador e Descodificador

A compressão de um sinal de vídeo é alcançada removendo do vídeo a redundância, isto é, componentes que não são necessárias para uma reprodução fiel do sinal de vídeo. Vários tipos de sinais de dados contêm redundâncias estatísticas e podem ser comprimidos usando algoritmos de compressão sem perdas (*lossless compression*). Com este tipo de algoritmos à saída do descodificador encontra-mos uma cópia exacta dos dados originais. Infelizmente este tipo de compressão é reduzida perante o que a indústria deseja, o melhor comportamento deste tipo de algoritmo dá origem ao formato JPEG-LS [16]. Neste formato consegue-se obter uma taxa máxima de compressão de 4 para 1, pouco para o desejado. Dito isto a indústria recorreu a algoritmos que perdem alguns dados da compressão para a descompressão mas que ao nível visual não é nada de significante. Os algoritmos com perdas (*lossy compression*) baseiam-se no princípio de remoção da redundância subjectiva (*subjective redundancy*), isto é, elementos que podem ser removidos da imagem sem alteração da percepção do utilizador/cliente que se encontra a visualizar a sequência de vídeo.

A maior parte dos sistemas de vídeo utiliza métodos que exploram tanto a redundância ao nível espacial como temporal, de forma a alcançar a compressão desejada.

No domínio temporal, existe uma correlação entre frames capturadas de forma consecutiva (a correlação entre frames consecutivas aumenta quanto maior for a taxa de amostragem), isto é, se imaginar uma bola preta a passar uma parede branca, a maior parte da imagem mantêm-se igual entre frames, é ai que entram os métodos de eliminação da redundância temporal, removendo toda essa informação que é igual em todas as frames e que estaria a ser transmitida desnecessariamente.

No domínio espacial existe uma elevada correlação entre píxeis próximos entre si, ai entra a redundância espacial aproveitando esse conhecimento de forma a comprimir a sequência de vídeo [5].

3.2 – Codificador de Vídeo

Um codificador de vídeo pode ser analisado como um conjunto de três grandes blocos. Estes são, o modelo temporal (*temporal model*), o modelo espacial (*espacial model*) e o codificador de entropia (*entropy encoder*) (Figura 18).

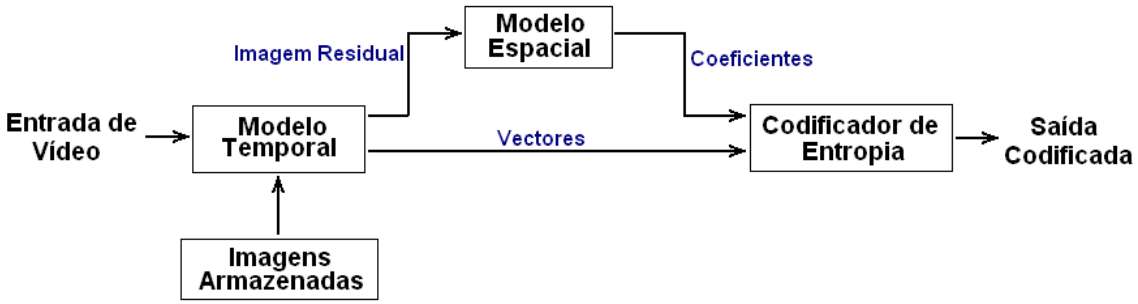

Figura 18- Modelo de um Codificador de Vídeo (adaptada de [5])

A entrada do modelo Temporal é uma sequência de vídeo por comprimir que realiza uma tentativa de redução da redundância temporal explorando as semelhanças entre frames consecutivas, construindo um modelo de preensão. No formato H.264 a predição é formada por uma ou mais frames antigas ou futuras e é melhorada compensando as diferenças (predição com compensação de movimento). A saída do modelo temporal é uma *frame* (imagem) residual criada da subtracção da *frame* actual com a sua predição e um conjunto de parâmetros de movimento. A saída do modelo Temporal descreve como o movimento é compensado (vectores de movimento).

A *frame* residual é utilizada como entrada do modelo espacial que usa as similaridades entre amostras vizinhas para reduzir a redundância espacial. No H.264 essa meta é alcançada aplicando uma transformada às amostras espaciais residuais e quantificando os resultados obtidos. A transformada converte as amostras noutro domínio no qual estas são representadas por coeficientes de transformada. Os coeficientes são posteriormente quantificados de forma a serem removidos os valores menos significativos, deixando assim um pequeno número de coeficientes significativos. Estes coeficientes tornam possível a representação da *frame* residual, de forma compacta. Assim sendo a saída do modelo espacial é um conjunto de coeficientes quantificados da transformada aplicada.

Os parâmetros de saída dos modelos Temporal e Espacial são então comprimidos num codificador de entropia que remove a redundância estatística presente na sequência de vídeo. A saída deste codificador apresenta então uma sequência de vídeo comprimida que pode ser armazenada ou transmitida (Figura 18).

Após se realizar a transmissão da sequência de vídeo ou mesmo o seu armazenamento, no momento de visualização, o descodificador reconstrói a imagem de vídeo a partir da sequência comprimida. Os coeficientes da transformada e os vectores de movimento são descodificados pelo descodificador de entropia, de seguida no modelo espacial os dados passam pelo descodificador do modelo espacial no qual é reconstruída a *frame* residual. O descodificador usa os vectores de movimento juntamente com uma ou mais *frames* residuais previamente descodificadas, criando a *frame* de predição. A *frame* actual é reconstruída então pela adição da *frame* residual à sua predição [5].

3.2.1 – Modelo Temporal

O modelo temporal apareceu com o grande objectivo de remover a redundância temporal das sequências de vídeo. Esta redução (compressão) dá-se devido à remoção de similaridades entre imagens consecutivas, realizando muitas das vezes predição da imagem actual do vídeo.

A saída do modelo temporal é uma *frame* residual proveniente da diferença da *frame* actual com a predita. A frame residual é codificada e enviada para o descodificador que regenera a *frame* actual. Neste processo de codificação e descodificação a precisão da imagem predita é tanto maior quanto melhor for o processo de compensação de movimento entre a *frame* predita e a *frame* actual.

1. Predição Proveniente da Imagem Anterior

O método mais simples de realizar predição de uma imagem é usar a imagem anterior como predição da imagem actual. Duas *frames* consecutivas (*frame* 1 e *frame* 2) de uma sequência de vídeo são mostradas na Figura 19, bem como a diferença entre as duas, a *frame* 1 é usada como predição da *frame* 2. Pela análise da imagem conseguida pela diferença entre as duas *frames*, podemos verificar que os pontos a cinzento mais claro correspondem a diferenças positivas e as zonas de cinzento mais escuro a diferenças negativas. A gama de cinzentos médios correspondem a zonas em que a diferença entre as imagens é nula, isto é, às duas *frames* coincidem.

Figura 19 - modelo temporal - predição utilizando a frame anterior (adaptada de [5])

Um grande problema deste método de predição é, como se pode ver pela imagem "Diferença entre *frames*" da figura anterior, a existência de muita quantidade de informação presente na *frame* residual, o que implica que existe muita informação a comprimir depois da predição temporal. Este método pode ser melhorado realizando uma estimação+compensação de movimento entre a *frame* de referência e a *frame* actual [5].

2. Mudanças Devido ao Movimento

Mudanças entre imagens de vídeo podem dever-se a diversos fenómenos. Grande parte das vezes devem-se ao movimento dos objectos presentes na sequência de vídeo, por exemplo uma bola que atravessa a tela de um lado ao outro. Podem advir também de movimentos da própria câmara que se encontra a capturar o momento, ser originárias de regiões da cena que se encontram sem protecção do objecto em movimento ou ainda ser originadas somente por variações de intensidade de luz aquando do registo da sequência de vídeo.

À excepção de zonas a descoberto e de variações de intensidade luminosa, as outras diferenças correspondem a movimentos de píxeis entre *frames*. Devido a

este facto torna-se possível estimar a trajectória de cada pixel entre *frames* sucessivas, produzindo um campo de trajectórias, conhecidas como fluxo óptico (do inglês *optical flow*) (Figura 20).

A Figura 20 apresenta um fluxo óptico da *frame* 1 presente na Figura 19. A imagem contém o campo completo de vectores de movimento para cada pixel que altera a sua posição. Desta técnica sabe-se que é possível realizar uma predição exacta para a maioria dos píxeis na *frame* em estudo, movendo cada pixel da sua referência pela direcção indicada pelo seu vector de fluxo óptico (vector de movimento). Contudo, este método de predição não é prático para a compensação de movimento dado que quanto maior precisão se quiser obter, mais cálculos terão de ser realizados utilizando a via computacional. Quanto maior a quantidade de informação a processar maior o tempo despendido a codificar e a descodificar, o que não é favorável para aplicações de vídeo em que o tempo que se leva a lançar um serviço de vídeo é importante.

Figura 20 - Fluxo Óptico (adaptada de [5])

3. Estimação e Compensação de Movimento Baseada em Blocos

A compensação de movimento baseada em blocos da *frame* em análise é um método muito usado actualmente. Esta forma de compensar segue sempre um método para cada bloco M x N da imagem.

Método de Compensação por Blocos de dimensão M x N:

1) Procurar na imagem de referência uma área de forma a realizar uma comparação exacta com o bloco de dimensão M x N da região amostra, isto é, realizando uma comparação do bloco M x N por todas as regiões possíveis da *frame* em análise. A forma mais popular de identificação da

zona da amostra é a realização da subtracção da amostra pela zona que nos encontramos a comparar. O bloco que tiver a menor energia residual é considerado a melhor escolha para a zona. Este processo de encontrar a melhor correspondência de bloco é conhecido como estimação de movimento (*motion estimation*).

2) A zona relativa ao bloco cuja correspondência foi a mais exacta, torna-se a amostra de predição para o bloco M x N corrente. Este é posteriormente subtraído pelo bloco corrente de forma a criar um bloco residual M x N, processo conhecido como compensação de movimento (*motion compensation*).

3) O bloco residual é codificado e transmitido, bem como a diferença entre o bloco corrente e a posição do bloco com a máxima correspondência (vector de movimento do inglês *motion vector*).

O descodificador recebe o vector de movimento de forma a recriar a região onde a predição se realizou e descodifica de seguida o bloco residual que também foi transmitido, adicionando-o à zona onde se realizou a predição. Este processo realiza a reconstrução do bloco original da imagem em estudo.

O método de reconstrução é muito popular, dado que é um método simples e computacionalmente tratável. Encaixa muito bem com imagens de vídeo rectangulares e com blocos baseados em transformadas, por exemplo, Transformadas Discretas de Co-senos muito conhecidas como DCT, derivadas do inglês *Discrete Cosine Transform.* O método de reconstrução produz resultados razoáveis para o modelo temporal, contudo, apresenta alguns problemas no que se trata à representação de objectos não rectangulares, objectos deformáveis e de movimento complexo. Estes problemas, devem-se à dificuldade que existe em encaixar os objectos irregulares em blocos de forma rectangular. Apesar deste inconveniente este método é o mais utilizado nos actuais padrões de codificação de vídeo (*video coding*) para transmissão e armazenamento.

4. Predição de Compensação de Movimento para Um Macrobloco

Um macrobloco (*macroblock*) corresponde a uma região de píxeis do tipo 16 x 16 de uma *frame* de vídeo. Este formato é comum em um número considerável de sistemas de codificação nos quais se encontram o MPEG-1, MPEG-2, MPEG-4 Visual, H.261, H.263 e H.264.

Para formatos de vídeo do tipo 4:2:0, a região de 16 x 16 do vídeo original é representada por 256 amostras de luminância (divididas por quatro blocos de 8 x 8), 64 amostras de componente azul de crominância (um bloco de 8 x 8) e 64 amostras de crominância vermelha (um bloco de 8 x 8), dando um saldo total de 6 blocos de dimensão 8 x 8 (Figura 21).

Um CODEC de MPEG-4 Visual ou H.264 processa cada *frame* de vídeo em unidades de macrobloco [5].

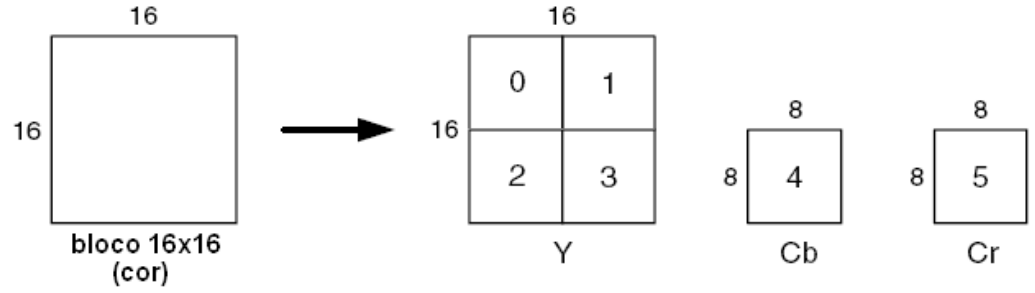

Figura 21 - Macrobloco (4:2:0) (adaptada de [5])

Estimação de Movimento:

Estimar o movimento de um macrobloco (MB) envolve encontrar uma zona na *frame* de referência de dimensão 16 x 16 que sirva de amostra, essa zona tem de ser semelhante à presente no macrobloco actual. A *frame* referência é uma *frame* da sequência de vídeo previamente codificada.

Compensação de Movimento:

A região seleccionada na qual se dá a melhor correspondência na *frame* de referência é subtraída (região retirada) ao macrobloco actual de forma a produzir um macrobloco residual (luminância e crominância) que é posteriormente codificado e transmitido em conjunto com o vector de movimento. O referido vector de movimento descreve a posição do macrobloco que tem a melhor correspondência (relativamente à posição do macrobloco em estudo).

Dentro do codificador o bloco residual é codificado, descodificado e adicionado à zona na qual se deu a correspondência máxima de forma a reconstruir o macrobloco original. De seguida o macrobloco é armazenado como referência para a próxima predição com compensação de movimento.

Para garantir que o codificador e o descodificador usem uma *frame* de referência idêntica para a compensação de movimento, torna-se necessária a utilização de um descodificador residual na reconstrução do macrobloco.

5. Compensação de Movimento - Tamanho do Bloco

Considerando as duas *frames* consecutivas de uma sequência de vídeo, presentes no Anexo 4. A *frame* 1 é subtraída à *frame* 2 sem compensação de movimento para produzir uma frame residual ("*Frame* residual Sem Compensação de movimento").

Ao se introduzir compensação de movimento com um bloco de tamanho 16 x 16, podemos logo à partida, verificar que a quantidade de energia presente na frame residual diminui drasticamente em comparação com o caso anterior ("*Frame* residual com bloco de 16 x 16").

Com compensação de movimento com bloco de dimensão 8 x 8 ("*Frame* residual com bloco de 8 x 8") verifica-se que, em comparação com a compensação de movimento com bloco de 16 x 16, a quantidade de energia diminui ainda mais. O mesmo se denota quando se realiza compensação de movimento com bloco de dimensão 4 x 4 ("*Frame* residual com bloco de 4 x 4"). Este estudo demonstra que

blocos de menor dimensão produzem melhores resultados no que se trata à compensação de movimento. Contudo, quanto menor a dimensão do bloco, maior a complexidade do sistema, maior a quantidade de processos de busca de compatibilidade de macrobloco, bem como maior a quantidade de vectores de movimento. Implicando assim uma maior quantidade de bits a serem transmitidos.

A escolha do tamanho de bloco ideal depende sempre da aplicação desejada bem como dos recursos (computacionais, monetários e largura de banda) disponíveis para a mesma.

A Norma MPEG-4 AVC/H.264 recorre a um algoritmo adaptativo de selecção do tamanho de bloco para realizar uma melhor compensação de movimento – blocos de tamanho variável.

6. Compensação de Movimento - Sub-pixel

Em cenas (de vídeo) no qual ocorrem movimentos de objectos, é normal que a deslocação do objecto não se dê entre píxeis, isto é, o movimento regista-se ao nível da fracção do pixel da imagem. Nestas situações, uma melhor predição pode ser obtida procedendo à interpolação da imagem referência ao nível do sub-pixel, antes de realizar a pesquisa pelo melhor candidato. Para a resolução destas situações e de forma a alcançar uma melhor estimação do movimento, recorre-se a um algoritmo capaz de estimar o movimento ao nível do sub-pixel. Este algoritmo funciona da seguinte maneira:

a. Pesquisa do melhor candidato sem uso da resolução sub-pixel;
b. Para a zona definida pelo melhor candidato anterior, pesquisar por um novo e melhor candidato, mas agora com uma resolução de metade de pixel (*half-pixel*) – dobro da resolução original;
c. Após obtenção de um candidato melhor, realizar nova pesquisa por um candidato melhor. Nesta fase aumenta-se a resolução para uma quarta parte de pixel (*quarter-pixel*) – quatro vezes maior que a original.
d. De modo a aumentar cada vez mais a precisão, e se achar necessário, realizar nova pesquisa por um melhor candidato, aumentando para tal a resolução para um oitavo de pixel (*eight-pixel*) – oito vezes maior que o original.
e. O melhor candidato que será utilizado no final do processo iterativo é subtraído ao macrobloco corrente.

Na implementação deste método, quanto maior for o nível de interpolação realizado ao sistema, melhor será a compensação de movimento, é claro que quanto maior for o nível de interpolação realizado maior será a quantidade de informação a processar, implica ainda uma menor taxa de compressão.

7. Compensação de Movimento Baseada na Região

Os objectos em movimento presentes nas cenas de vídeo real, encontram-se raramente perfeitamente alinhados ao longo das fronteiras do bloco. O mais comum é o aparecimento de formas irregulares localizadas arbitrariamente. Este problema é

ilustrado pela Figura 22, na qual a forma oval é um objecto em movimento e a forma rectangular um objecto estático.

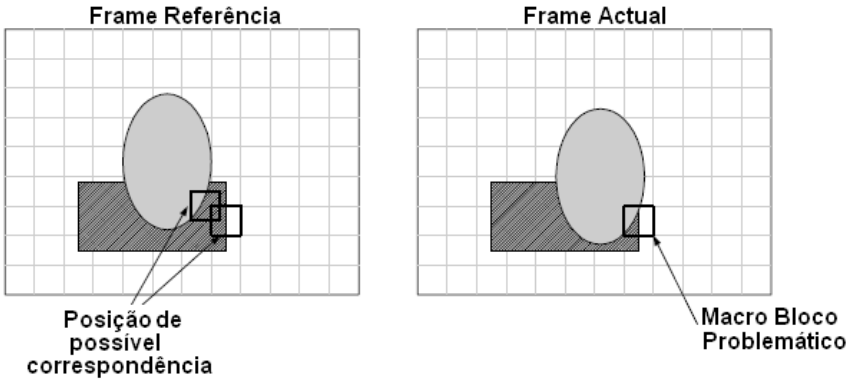

Figura 22 - Compensação de movimento de formas arbitrárias em objectos em movimento (adaptada de [5])

Em situações deste género é difícil encontrar uma boa correspondência na *frame* de referência para o macrobloco assinalado, pois abrange parte do objecto em movimento e parte do objecto estático. Nenhuma das posições possíveis para a correspondência, na *frame* referência, realiza uma correspondência ideal.

Uma melhor performance pode ser alcançada realizando compensação de movimento aleatoriamente às diversas regiões da imagem (*region-based compensation*). Por exemplo se tentarmos realizar apenas compensação de movimento para píxeis dentro da região oval, então é possível encontrar uma correspondência ideal na frame de referência. A compensação de movimento aleatório tem algumas dificuldades associadas a serem ultrapassadas para realizar uma compensação de movimento baseada na região (do macrobloco). Por exemplo, pode-se mencionar a identificação precisa e concisa (segmentação) das regiões de fronteira, o limite para a descodificação e codificação do movimento residual após compensação.

O MPEG-4 inclui um conjunto de ferramentas que apoiam a compensação de movimento baseada na região em estudo.

3.2.2 – Modelo Espacial

O modelo espacial ou modelo de imagem (*image model*), tem como função diminuir a correlação entre imagens vizinhas, isto é, decompor as imagens de forma a se poder realizar uma melhor compressão da imagem residual e transformar a imagem residual em informação facilmente comprimida pelo codificador de entropia.

As imagens de uma cena natural possuem um elevado nível de correlação entre amostras vizinhas da imagem, tornando assim difícil a sua compressão.

Realizando a auto-correlação bidimensional (2D) entre um pixel e um outro deslocado de algumas posições, verifica-se que quando um pixel se desloca de uma posição para a outra o declive da correlação não é acentuado. Este facto demonstra a correlação entre amostras adjacentes. Em contrapartida, ao se obter a resposta à função de correlação 2D da imagem residual, na qual se realizou compensação de movimento, o declive da curva é muito acentuado, ou seja, amostras adjacentes são pouco relacionadas (Figura 23).

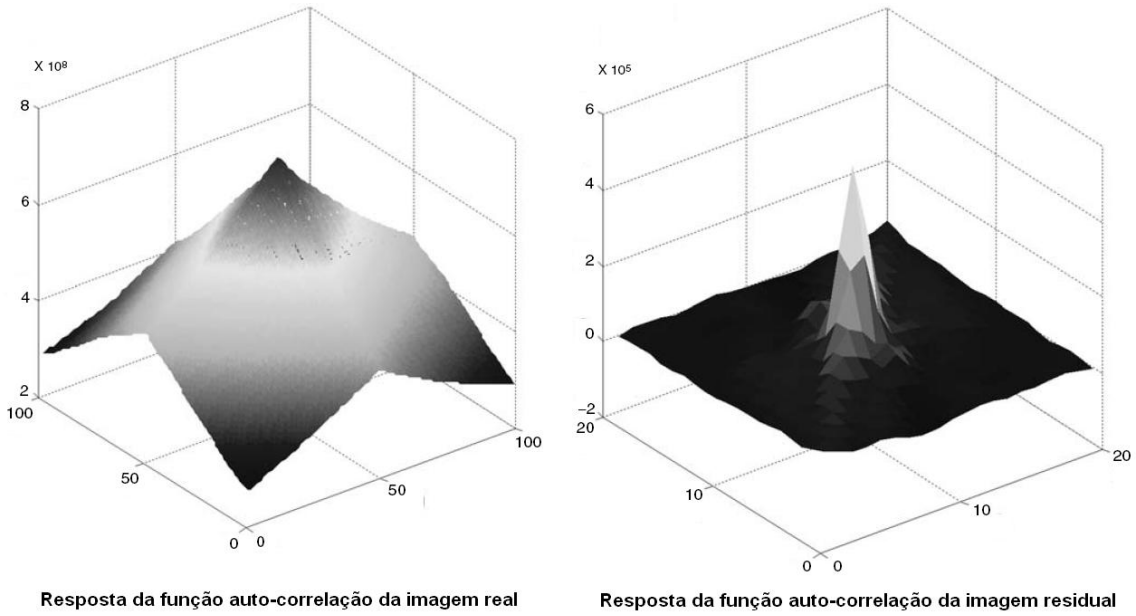

Figura 23 - Desempenho da função auto-correlação da imagem real e residual (adaptada de [5])

Normalmente o modelo espacial é composto por três componentes principais, a **transformada** (que descorrelaciona e compacta a informação), o **quantificador** (que reduz a precisão dos dados provenientes da transformada) e por fim o **ordenador** (que arranja os dados).

1. Transformada

A transformada num CoDec de vídeo converte os dados, informação, da imagem residual num domínio diferente (o domínio da transformada). A escolha da transformada a utilizar depende de um número de critérios abaixo enumerados:

i. Os dados no domínio da transformada devem encontrar-se separados por componentes e não podem ser correlacionados. Devem-se encontrar compactados, isto é, com a maior parte da energia concentrada em poucos valores;
ii. A transformada deve ser reversível;
iii. De forma a minimizar o tempo dispendido no processo de implementação a transformada escolhida deve ter a possibilidade de se representar de forma computacional.

A transformada mais comum ao nível industrial é a transformada discreta de co-senos (DCT – *Discrete Cosine Transform*) [17]. Esta transformada opera sobre blocos de dimensão N x N da imagem residual, o que implica que a imagem vai ser processada por blocos da mesma dimensão.

Este tipo de transformada é adequado para realizar a compressão da compensação de movimento da imagem residual por blocos. Por outro lado a transformada tem o inconveniente de tender a criar o aparecimento de "artefactos" nas fronteiras dos blocos analisados ("*blockiness*").

Para além da DCT existem ainda transformadas que trabalham sobre toda a imagem, como é o caso da *Discrete Wavelet Transform* (DWT) [17]. A DWT é usada num dos perfis do CoDec do MPEG-4 Visual, porém, para operar em condições tem necessidade de uma elevada quantidade de recursos ao nível da memória. A elevada quantidade de memória deve-se à necessidade de processar a *frame* por um todo, em vez de blocos como é o caso da DCT. Por esta razão este tipo de transformada não é a mais adequada a compensações de movimento por blocos [5].

I. Transforma Discreta de Co-Senos

A DCT opera com blocos de dimensão **X**, em que **X** corresponde à dimensão N x N, tipicamente blocos da imagem ou da imagem residual após predição e cria **Y** que corresponde a um bloco de coeficientes de dimensão N x N. A acção realizada pela DCT e pela sua inversa, IDCT – *Inverse Discrete Cosine Transform* pode ser descrita pelos termos da transformada da matriz **A**.

A transformada de um bloco de dimensão N x N pode ser obtida por [5]:

$$Y = AXA^T \tag{3.1}$$

e a sua inversa (IDCT) por:

$$X = A^T YA \tag{3.2}$$

onde **X** é a matriz das amostras, **Y** a matriz dos coeficientes e **A** é a matriz de dimensão N x N da transformada. Os elementos de A são:

$$A_{ij} = C_i cos\frac{(2j+1)i\pi}{2N} \quad \text{Onde} \quad C_i = \begin{cases} \sqrt{\frac{1}{N}} & , i = 0 \\ \sqrt{\frac{2}{N}} & , i > 0 \end{cases} \tag{3.3}$$

As equações 3.1 e 3.2 podem ainda ser descritas segundo a forma:

$$Y_{xy} = C_x C_y \sum_{i=o}^{N-1} \sum_{j=0}^{N-1} X_{ij} cos\frac{(2j+1)y\pi}{2N} cos\frac{(2i+1)x\pi}{2N} \tag{3.4}$$

$$X_{ij} = \sum_{x=o}^{N-1} \sum_{y=0}^{N-1} C_x C_y Y_{xy} cos\frac{(2j+1)y\pi}{2N} cos\frac{(2i+1)x\pi}{2N} \tag{3.5}$$

Exemplo: *Desenvolvimento da matriz transformada A para um N=4*

A matriz transformada A, para um N=4 apresenta os seguintes coeficientes

$$A = \begin{bmatrix} \frac{1}{2}\cos(0) & \frac{1}{2}\cos(0) & \frac{1}{2}\cos(0) & \frac{1}{2}\cos(0) \\ \sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{5\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{7\pi}{8}\right) \\ \sqrt{\frac{1}{2}}\cos\left(\frac{2\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{6\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{10\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{14\pi}{8}\right) \\ \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{9\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{15\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{21\pi}{8}\right) \end{bmatrix} \quad (3.6)$$

Dadas as propriedades de simetria e de repetição que o co-seno apresenta a cada 2π radianos, a matriz de transformação **A** pode ser simplificada, ficando:

$$A = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) & -\sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) & -\sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right) \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) & -\sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right) & -\sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) \end{bmatrix} \quad (3.7)$$

Se $a = \frac{1}{2}$, $b = \sqrt{\frac{1}{2}}cos\left(\frac{\pi}{8}\right)$ e $c = \sqrt{\frac{1}{2}}cos\left(\frac{3\pi}{8}\right)$ então a matriz de transformação,

$$A = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \quad (3.8)$$

Ao se calcular os valores correspondentes a cada posição da matriz de transformação **A** ficamos com:

$$A = \begin{bmatrix} 0{,}5000 & 0{,}5000 & 0{,}5000 & 0{,}5000 \\ 0{,}6533 & 0{,}2706 & -0{,}2706 & -0{,}6533 \\ 0{,}5000 & -0{,}5000 & -0{,}5000 & 0{,}5000 \\ 0{,}2706 & -0{,}6533 & 0{,}6533 & -0{,}2706 \end{bmatrix} \quad (3.9)$$

A saída de uma transformada bidimensional de co-senos (DCT) é um conjunto coeficientes de dimensão N x N, que representam o bloco de dados da imagem no domínio da transformada. Estes coeficientes podem ainda ser considerados como os "pesos" para o padrão base da imagem. O padrão de base de decomposição para a DCT de dimensão 4 x 4 e 8 x 8 é apresentado no Anexo 5.

Qualquer bloco de imagem pode ser reconstruído combinando todos os parâmetros N x N do padrão base e multiplicando pelo factor base adequado (respectivo coeficiente da transformada).

Exemplo: *Cálculo da DCT de um bloco de amostras de dimensão 4 x 4*

O bloco seguinte (X) é um bloco de amostras retirado de uma imagem de vídeo:

	j=0	*1*	*2*	*3*
i=0	5	11	8	10
1	9	8	4	12
2	1	10	11	4
3	19	6	15	7

Para o cálculo da matriz de saída da DCT do bloco de amostras, basta aplicar a equação 3.1

$$Y = AXA^T$$

$$= \begin{bmatrix} 0{,}5000 & 0{,}5000 & 0{,}5000 & 0{,}5000 \\ 0{,}6533 & 0{,}2706 & -0{,}2706 & -0{,}6533 \\ 0{,}5000 & -0{,}5000 & -0{,}5000 & 0{,}5000 \\ 0{,}2706 & -0{,}6533 & 0{,}6533 & -0{,}2706 \end{bmatrix} \begin{bmatrix} 5 & 11 & 8 & 10 \\ 9 & 8 & 4 & 12 \\ 1 & 10 & 11 & 4 \\ 19 & 6 & 15 & 7 \end{bmatrix} \begin{bmatrix} 0{,}5000 & 0{,}5000 & 0{,}5000 & 0{,}5000 \\ 0{,}6533 & 0{,}2706 & -0{,}2706 & -0{,}6533 \\ 0{,}5000 & -0{,}5000 & -0{,}5000 & 0{,}5000 \\ 0{,}2706 & -0{,}6533 & 0{,}6533 & -0{,}2706 \end{bmatrix}^T$$

$$= \begin{bmatrix} 35{,}0000 & -0{,}0793 & -1{,}5000 & 1.1152 \\ -3{,}2992 & -4{,}7678 & 0{,}4427 & -9{,}0104 \\ 5{,}5000 & 3{,}0286 & 2{,}0000 & 4{,}6987 \\ -4{,}0454 & -3{,}0104 & -9{,}3837 & -1{,}2322 \end{bmatrix}$$

Exemplo: *Bloco de Imagem e Coeficientes da DCT*

A Figura 24 mostra uma imagem segmentada por blocos de dimensão 4 x 4 (a caixa a branco representa um bloco de dimensão 4 x 4). Na Figura 25 apresenta-se um zoom a um bloco da imagem apresentada na Figura 24. À partida não se compreende o porquê de se utilizar a transformada de Co-senos uma vez que não existe redução da quantidade de dados. Por exemplo, para 16 valores de pixel, necessitamos de 16 coeficientes de DCT. A vantagem que se verifica ao utilizar a DCT torna-se clara quando o bloco de imagem é reconstruído a partir de um subconjunto de coeficientes.

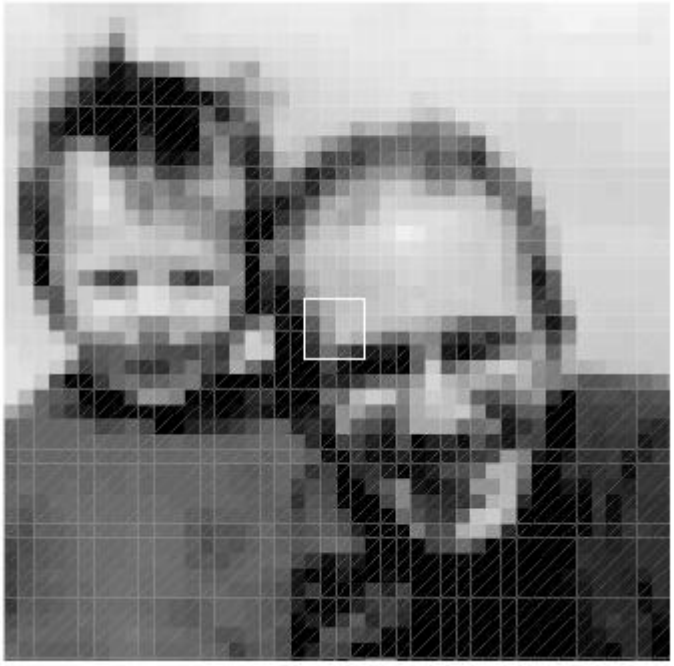

Figura 24 - Segmentação da Imagem em blocos de dimensão 4 x 4 (adaptada de [5])

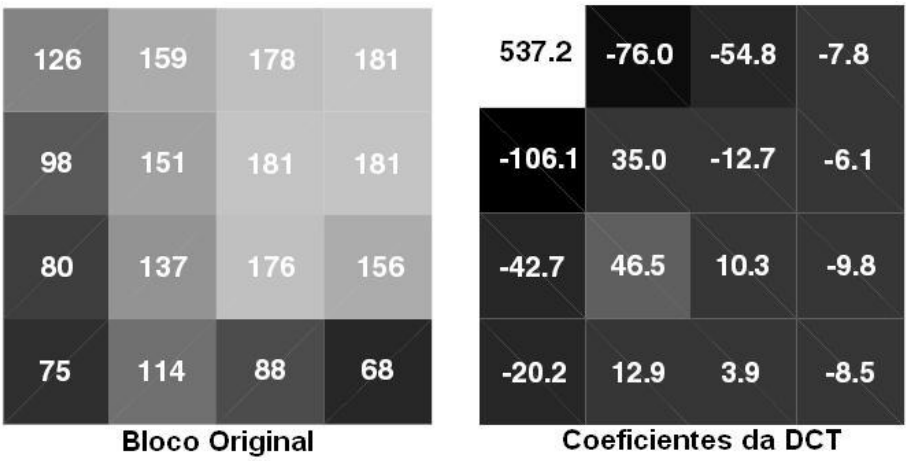

Figura 25 - Zoom a um bloco de dimensão 4 x 4 e respectiva representação em coeficientes de DCT

No processo inverso da transformada – IDCT, colocam-se todos os coeficientes a zero excepto os mais significativos. Na realização deste processo (IDCT) obtém-se a saída do bloco mostrado no Anexo 6 (ii. *a*), no qual se pode verificar que para todas as posições se encontra a média dos valores originais de pixel. Calculando a IDCT das duas componentes mais importantes obtém-se o bloco reconstruído como se encontra no Anexo 6 (ii. *b*). À medida que se adicionam mais coeficientes ao cálculo da IDCT, melhor e mais rigorosa é a reconstrução da imagem inicial (Anexo 6 (ii. *c*) e Anexo 6 (ii. *d*)). Este tratamento é um "jogo" que permite obter uma imagem aproximadamente igual da original. Deste modo torna-se possível obter uma imagem semelhante á original a partir de um subconjunto de 16 coeficientes da DCT, Anexo 6 (i.).

Ao remover os coeficientes de menor amplitude da IDCT, como por exemplo por quantificação, permite-nos que os dados da imagem possam ser representados com um número reduzido de coeficientes. Claro que a remoção de informação resultará sempre em perdas, logo perda de resolução na imagem final, o que implica perda de qualidade [5][17].

II. *Wavelet Transform*

A *wavelet transform*, muito usada na compressão de imagem, é baseada num conjunto de filtros com coeficientes que são equivalentes a funções *discrete wavelet transforms*. Uma função *discrete wavelet transform* parte de um sinal contendo N amostras. Um par de filtros é aplicado ao sinal decompondo-o em duas componentes, uma para a banda das baixas frequências (L) e outra para a banda das altas frequências (H). De seguida em cada banda é reduzido o número de amostras presentes para análise, dividindo as amostras (*down-sample*) por um factor de 2. Assim cada banda fica dividida em dois, criando novas bandas de frequência, contendo cada nova banda N/2 amostras. Com este tratamento e com a escolha dos filtros correctos a operação de divisão é reversível.

Esta abordagem pode ser aplicada a um sinal bidimensional, como é o caso de uma *frame* de vídeo. Cada linha da imagem 2D é filtrada com um filtro passa-baixo e com um passa-alto (L_x e H_x). À saída de cada filtro é realizada uma divisão do número de amostras (*down-sampled*) por um factor de dois, de forma a produzir as imagens intermédias L e H. L é a imagem original filtrada com passa-baixo e com redução de amostras (*down-sample*) na direcção do eixo x. H é a imagem original filtrada por um filtro passa-alto na direcção do eixo x e com diminuição de amostras (*down-sample*). De seguida cada coluna das novas imagens é filtrada novamente com filtros passa-alto e passa-baixo (L_y e H_y) sendo divididas novamente por um factor de dois (*down-sampled*). Este processo de amostragem produz quatro sub imagens finais (ou sub bandas), LL, LH, HL e HH. As sub imagens finais podem ser combinadas para se obter como imagem de saída uma imagem semelhante à inicial (*frame* original) com o mesmo número de amostras que a original. No conjunto das quatro sub imagens produzidas encontra-se toda a informação da imagem original.

Como as sub bandas são de natureza escassa, isto é, contêm pouca quantidade de informação, faz com que estas se tornem passíveis de compressão.

Quando se trata de aplicações para a compressão de imagem, torna-se necessário remover a informação redundante, retirando-se amostras próximas de zero ou mesmo zeros que em nada contribuem para a transmissão da imagem.

No descodificador, a imagem é reconstruída realizando repetitivamente *up-sampling*, filtragem e adição de coeficientes [5][17].

2. Quantificador

Um quantificador "mapeia" um sinal com uma amplitude **X** num sinal quantificado com um número reduzido de valores **Y**.

O quantificador pode ser dividido em dois modos de operar, pode ser do tipo numérico (*scalar quantiser*), em que uma amostra do sinal de entrada corresponde a uma amostra na saída do sinal quantificado, ou pode ser do tipo vectorial (*vector*

quantiser), em que um número de amostras do sinal de entrada (um vector) corresponde a um conjunto de valores quantificados na saída.

A. Quantificador Numérico (*scalar quantiser*)

O modo mais simples de explicar este tipo de quantificador é dar um exemplo, assim sendo, comecemos por pensar no caso em que temos um número decimal com algumas casas decimais. Ao se proceder à aproximação deste número ao inteiro mais próximo, estamos a pegar num sinal X na entrada e a quantificá-lo. À amostra na entrada X, estamos a corresponder com uma amostra quantificada na saída, Y. Este processo tem perdas e não é reversível, dado que deixa de ser possível determinar o valor decimal à entrada do quantificador a partir do valor aproximado.

Um quantificador geral pode ser definido da seguinte forma:

$$FQ = round\left(\frac{X}{QP}\right)$$
$$Y = FQ.QP \tag{3.10}$$

Em que *QP* é o tamanho do degrau (passo) de quantificação. Todos os valores quantificados encontram-se espaçados uniformemente em intervalos de *QP*.

Dentro da categoria dos quantificadores numéricos existem dois tipos diferentes, os lineares e os não lineares (Figura 26).

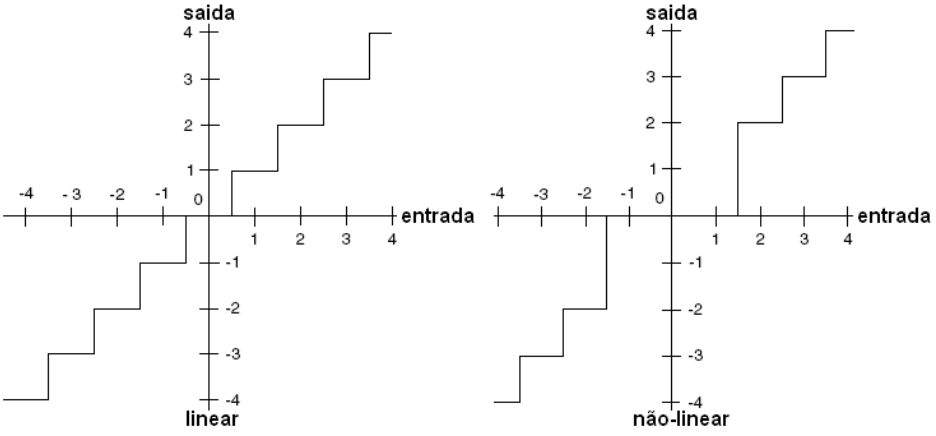

Figura 26 - Tipos de Quantificador Numérico (adaptada de [5])

Em CoDecs de vídeo, o processo de quantificação é realizado em duas fases: um quantificador para a frente (FQ) (*forward quantiser*) e um quantificador inverso (IQ) (*inverse quantiser*) [18][5]. Uma vez que quantificar não ser é processo reversível, é mais correcto chamar ao FQ de escalonador (*scaler*) e ao IQ de repetidor de escalonamento (*rescaler*).

Da análise ao quantificador pode-se concluir que quanto maior for o valor de QP maior será a compressão obtida. Mas, comprimir significa perder qualidade, significa que se irá obter uma imagem de pior qualidade quanto maior for a taxa de compressão. A escolha de QP leva a uma escolha entre a quantidade de perdas que se pretende ter e a qualidade de imagem desejada na aplicação final.

O FQ deve ser desenhado para que converta valores insignificantes da DCT para zero, enquanto mantém em simultâneo um número reduzido de coeficientes significativos. A típica saída de um quantificador bem projectado é um conjunto de valores (*array*), contendo maioritariamente zeros e com alguns valores diferentes de zero, a estes dá-se o nome de coeficientes da imagem.

B. Quantificador Vectorial (*vector quantiser*)

Um quantificador vectorial converte um conjunto de dados de entrada para um único valor, como é o caso de um bloco amostra de uma imagem. À posterior, no descodificador esse valor corresponde a uma aproximação dos valores originais (um "vector"). O conjunto de vectores é armazenado no codificador e descodificador numa "tabela" (*codebook*).

Um modo de aplicação deste tipo de quantificador é:

a) Dividir a imagem original em regiões (blocos);
b) Escolher um vector da tabela de vectores que se assemelhe o mais possível com a região escolhida;
c) Transmitir para o descodificador o índice que identifica o vector escolhido;
d) No descodificador, reconstruir uma cópia aproximada (visto que o processo de quantificação é um processo que tem perdas associadas) da região usando o índice fornecido (Figura 27).

Ao projectar um quantificador vectorial deve ser sempre tomado em consideração o tipo de *codebook*, bem como os métodos a usar na busca pelos vectores. Deve existir também o cuidado de procurar sempre utilizar métodos de pesquisa rápida, de forma a diminuir os tempos de processamento de cada imagem.

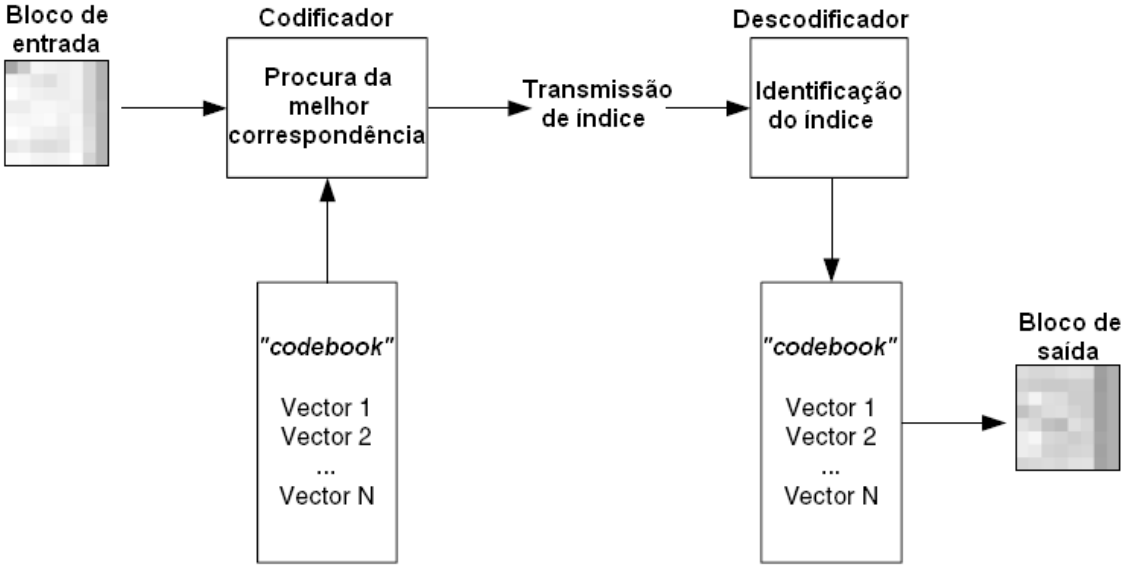

Figura 27 - Quantificador Vectorial (adaptada de [5])

3. Reordenação e Codificador de Zeros

Na saída do quantificador obtém-se um conjunto de valores ordenados, ou seja, um *array* que é constituído por poucos coeficientes diferentes de zero. Sendo maioritariamente constituído por zeros, torna-se então necessário proceder à reordenação do *array*. Agrupam-se os valores não nulos e representam-se de forma eficiente os coeficientes nulos [5].

A forma de reordenar os coeficientes não nulos depende muito da forma como se encontram ordenados os coeficientes não nulos da DCT. Os coeficientes mais significativos da DCT de um bloco encontram-se normalmente em redor da componente DC do bloco, ou seja, ocupam posições de baixa frequência. Ao ser calculada a probabilidade de não ser zero nos coeficientes da DCT de um bloco, verifica-se que existem relações de dependência entre a presença de coeficientes nãos nulos e a componente DC do bloco. Verifica-se ainda a existência de simetria ao nível vertical e horizontal.

Quando se analisa vídeo progressivo e entrelaçado a distribuição de coeficientes dá-se de forma diferente, em diagonal. Os coeficientes não nulos encontram-se mais à esquerda do gráfico de probabilidades da imagem. Isto deve-se á presença de uma componente mais elevada de alta frequência no eixo vertical, devido à realização de sub amostragem vertical.

Devido ao conhecimento destas características, realiza-se um reordenamento dos valores da DCT em ziguezague realizado segundo a *frame* (*ZigZag scan order - frame block*), após a realização da quantificação (Figura 28 *A)*).

Para vídeo entrelaçado, o processo de reordenamento dá-se segundo a forma apresentada na Figura 28 *B)*, consiste num varrimento realizado segundo o campo (*ZigZag scan order – field block*), dado que a primeira forma apresentada não tem um comportamento ideal nestas condições.

Após a realização do varrimento (processo de reordenação), é necessário realizar a codificação da sequência de coeficientes obtida, para representar de uma forma mais compacta o elevado número de zeros existente no *array* de coeficientes, a este processo dá-se o nome de *Run Level Encoding*.

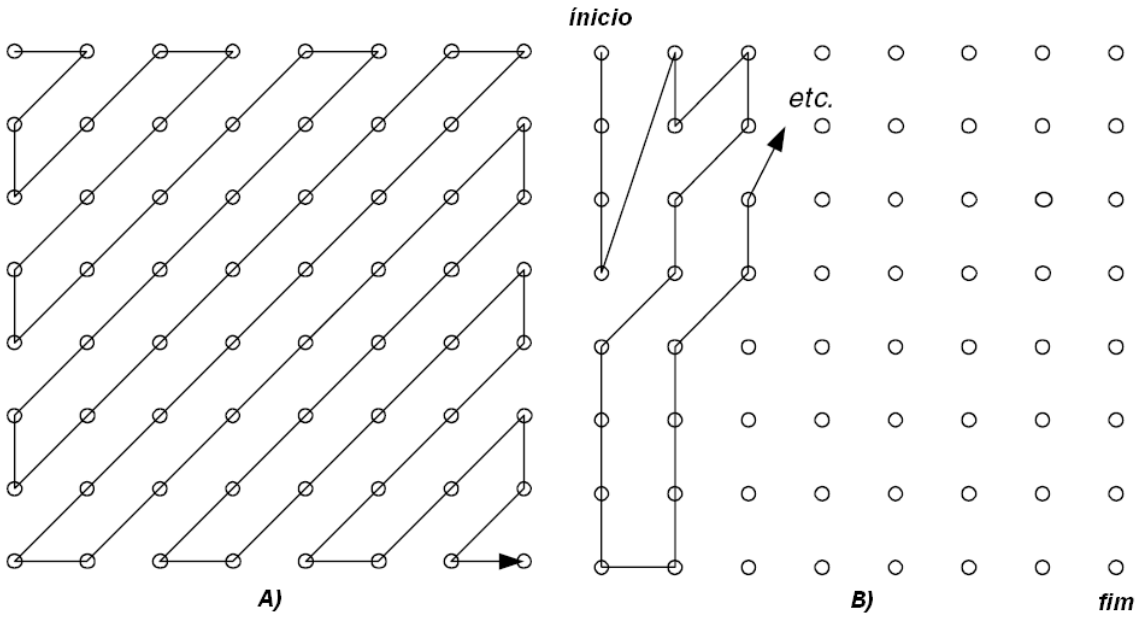

Figura 28 - Varrimento em Ziguezague para - *A) frame* e *B)* campo (adaptada de [5])

Para além do método de tratamento dos coeficientes da DCT existe ainda o método de tratamento para os coeficientes da transformada *wavelet*. Com a aplicação desta transformada ficamos com muitos dos coeficientes das sub bandas mais elevadas perto de zero, os quais podem ser quantificados para zero sem grandes perdas da qualidade da imagem. Em contrapartida, coeficientes diferentes de zero tendem a corresponder a estruturas na imagem.

Quando um coeficiente numa banda de baixa frequência é diferente de zero existe uma elevada probabilidade de que coeficientes na posição correspondente mas em altas frequências também o sejam. Dito isto, podemos considerar uma rede de coeficientes quantificados diferentes de zero, partindo da raiz numa banda de baixa frequência. É desejável para codificar os coeficientes diferentes de zero provenientes da transformada *wavelet*, que os coeficientes se encontrem da forma mais compacta possível antes de passarem pelo codificador de entropia. Uma forma de o conseguir é codificar a rede de coeficientes diferentes de zero, desde o nível mais baixo (raiz) de decomposição. Um coeficiente na camada mais baixa é codificado, seguido pelo nó mais próximo da rede e assim em diante. A codificação continua até que se atinja um coeficiente da rede cujo valor seja igual a zero. No descodificador realiza-se a descodificação partindo da raiz da rede, os coeficientes diferentes de zero são descodificados e reconstruídos, quando a rede se encontra descodificada, todos os nós restantes da rede são colocados a zero. Este método de codificar coeficientes da transformada wavelet é conhecido como *embedded zero tree* (EZW).

3.2.3 – Codificador de Entropia

A função do codificador de entropia é transformar uma série de símbolos que representam elementos de uma sequência de vídeo numa sequência de vídeo comprimida (*bitstream*) adequada para a transmissão ou mesmo para o armazenamento.

Num codificador de vídeo a série de símbolos, que representa a sequência de vídeo, normalmente é composta por coeficientes quantificados e reordenados da transformada, vectores de movimento, marcadores (usados para a sincronização de sequências de vídeo, quando existe perda de sincronismo), cabeçalhos (do macrobloco, da sequência de vídeo ou mesmo da *frame*) e outros tipos de informação não necessária para a correcta descodificação.

1. **Codificação Preditiva** – ***Predictive coding***

   A compressão de vectores de movimento pode ver o seu desempenho melhorado sempre que se realiza uma predição do vector de movimento baseada em vectores de movimento codificados anteriormente. Assim na transmissão apenas seria necessário o envio da informação alterada (a informação relativa à codificação diferença entre a predição e o vector de movimento anterior) em vez de se estar a enviar os vectores de movimento completos, a este processo dá-se o nome de **MVD – Diferença entre Vectores de Movimento**, do inglês *Motion Vector Difference*.

   Utilizando esta técnica existe uma melhoria de eficiência significativa, dado que existe uma menor quantidade de informação a circular na rede, isto é, existe uma compressão uma vez que são necessários menos bits para representar as diferenças do que seria necessário para transmitir a informação completa.

2. **Codificação de Comprimento Variável** – ***Variable-length Coding***

   Um codificador de comprimento variável (VLC) "mapeia" símbolos de entrada numa série de palavras-chave (*keywords*). Cada símbolo é mapeado por uma palavra-chave de tamanho variável, mas todas contêm um número de bits mínimo. As palavras-chave que ocorrem num maior número de vezes terão um comprimento de palavra menor (menos bits) do que a palavra-chave de um símbolo que ocorre muito poucas vezes. Este comportamento ajuda a que se dê compressão de dados visto que símbolos que se repetem mais vezes serão representados com menor quantidade de bits.

   - ***Códigos de Huffman***

     Nos Códigos de *Huffman* [19], é atribuído a cada símbolo um VLC de acordo com a probabilidade de ocorrência de símbolos diferentes. De acordo com *D. Huffman*, é necessário construir um conjunto de palavras-chave de códigos variáveis de acordo com os resultados do cálculo das probabilidades de ocorrência de cada símbolo. Se existir uma distribuição exacta das probabilidades, então obtém-se uma representação relativamente compacta dos

símbolos originais. Em contrapartida este tipo de representação, quando aplicada a CoDecs de vídeo, obriga a que o descodificador tenha acesso à tabela de códigos do codificador. Devido a este facto é necessária a transmissão de toda a tabela de probabilidades do codificador para o descodificador. A transmissão desta quantidade de dados adicionais provoca um aumento da carga de transacções na rede, reduzindo de certa forma a taxa de compressão obtida com a utilização desta técnica. A perda de desempenho acentua-se ainda mais quando as sequências de vídeo a transmitir são de dimensões reduzidas. Com a necessidade de transferir toda a tabela do codificador para o descodificador, verifica-se o aparecimento de um novo problema. A tabela de códigos só pode ser calculada quando o filme é totalmente codificado criando assim um atraso significativo na transmissão da sequência de vídeo. Atraso inaceitável para todo o processo de codificação da sequência.

De forma a resolver os problemas de codificação usando códigos de *Huffman*, entidades reguladoras relacionadas com a área da codificação, definiram "normas", definiram mais em concreto as "palavras código" (*codewords*), baseadas em comportamentos probabilísticos e estatísticos de conteúdos de vídeo genéricos. Dado que com este tipo de codificação é necessário previamente calcular as tabelas antes de se realizar a transmissão, então basta armazenar estas tabelas logo à partida no codificador e no descodificador.

Uma desvantagem marcante da codificação de *Huffman* e de codificações baseadas em códigos de *Huffman* em detrimento de outros modelos de codificação do tipo VLC é o problema proveniente da transmissão de dados, isto é, se existir uma perda de um pacote da sequência de vídeo na transmissão do codificador para o descodificador mesmo que este seja muito pequeno, implica na perda imediata de sincronismo, falhando assim a descodificação dos códigos consecutivos ao erro. Por esta razão criaram-se os **códigos de comprimento variável reversíveis** (RVLC), que permitem descodificação em ambas as direcções o que aumenta a imunidade do sistema a erros [5].

- ***Códigos Aritméticos***

Com códigos VLC pré calculados, torna-se necessário o aumento da quantidade de memória disponível para armazenamento das tabelas pré calculadas, tanto no codificador como no descodificador. Ao serem utilizados códigos aritméticos geram-se automaticamente códigos por cada símbolo que entra no codificador. Para além disso, a codificação aritmética é "inteligente", isto é, o comprimento dos códigos gerados depende do conteúdo do símbolo que deu entrada. Ao contrário do que acontece nos sistemas VLC. Um codificador aritmético converte uma sequência de símbolos num único número fraccionário conseguindo aproximar-se do número óptimo de bits para representar cada símbolo [5].

Na norma MPEG-4 AVC/H.264, são utilizados códigos aritméticos do tipo Golomb exponencial, *Exp-Golomb*. Nas mais actuais aplicações de MPEG-4 Visual e MPEG-4 AVC/H.264, a codificação aritmética usada é baseada no contexto CAE (*Context-based Arithmetic Encoding*), isto é, consiste em aproveitar o conhecimento das características espaciais e temporais para estimar as probabilidades de cada símbolo a codificar.

3.3 – Módulo de Compatibilidade com Versões Anteriores – DPCM/DCT

A maioria dos principais codificadores anteriores a 1990 baseia-se no modelo híbrido DPCM/DCT. Este modelo incorpora um bloco de estimação e compensação de movimento (DPCM), um bloco de transformada (DCT) e um módulo com um codificador de entropia (Figura 29 e Figura 30).

Cada CoDec que se diga compatível com H.261, H.263, MPEG-1, MPEG-2, MPEG-4 Visual e H.264 tem de ter incorporado um conjunto de funções de codificação e descodificação que implemente este modelo DPCM/DCT.

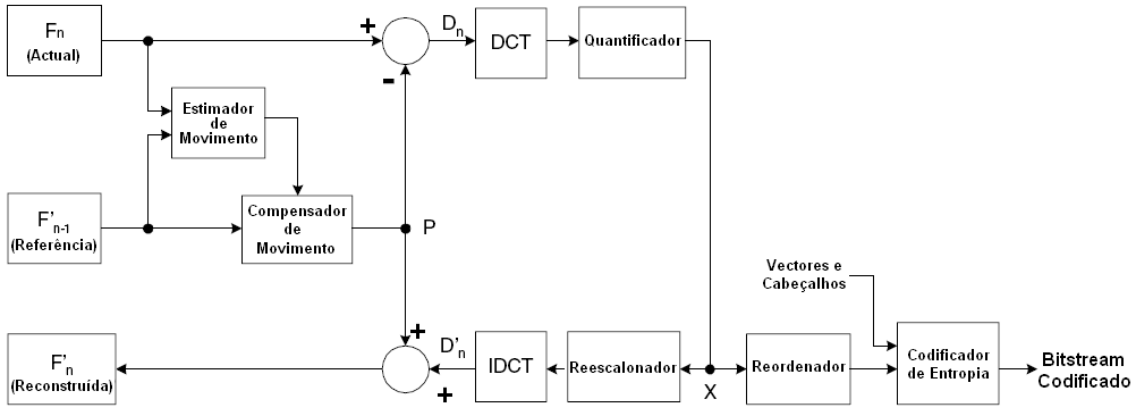

Figura 29 - Codificador de vídeo DPCM/DCT (adaptada de [5])

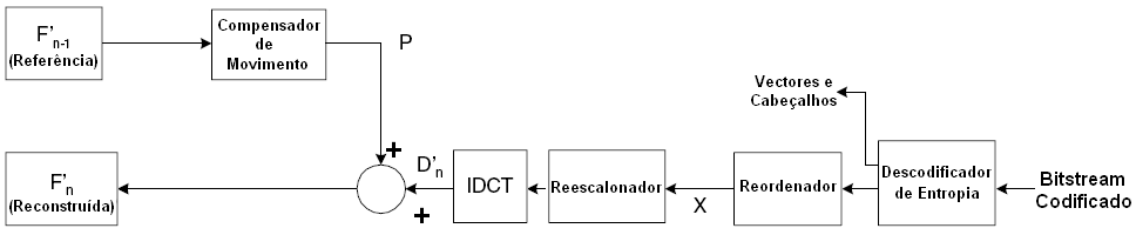

Figura 30 - Descodificador de vídeo DPCM/DCT (adaptada de [5])

Nas figuras anteriores podemos verificar um codificador e um descodificador DPCM/DCT. Como se pode verificar o descodificador é muito semelhante ao codificador. Observa-se ainda que o codificador possui na sua configuração uma malha (*loop*) de descodificação, tal deve-se ao facto do codificador necessitar de possuir uma referência para codificar as imagens que o descodificador vai usar.

Modo de funcionamento do codificador:

1. A imagem de vídeo (*frame*) F_n, que vai ser codificada é dividida em macroblocos. Estes na norma MPEG-4 AVC/H.264 apresentam um tamanho de bloco fixo de 16 x 16 para a luminância e 8 x 8 para cada componente de crominância.
2. F_n é comparada com uma imagem de referência que pode ser uma imagem previamente codificada (F_{n-1}). Então o descodificador de movimento realiza uma pesquisa por uma região de dimensão 16 x 16 em F_{n-1}, ou numa interpolação de F_{n-1}, com o intuito de identificar o macrobloco que melhor se adapte ao macrobloco actual de F_n. O desvio entre o macrobloco encontrado e o macrobloco actual dá o vector de movimento (MV).
3. Com base no vector de movimento, uma predição *P* é gerada. *P* é a região de 16 x 16 seleccionada pelo estimador de movimento.
4. Ao macrobloco actual subtrai-se *P* de forma a se obter o macrobloco residual *D*.
5. *D* é passado pelo bloco de transformada, ao qual é aplicado a transformada DCT, tipicamente *D* é ainda subdividido em sub blocos de dimensão 8 x 8 e 4 x 4, os quais são transformados individualmente.
6. Cada sub bloco é quantificado, originando *X*.
7. Por cada coeficiente da DCT dá-se a reordenação e codificação pelo método de *Run-Level*.
8. Finalmente, os coeficientes (vector de movimento e cabeçalhos associados) são codificados pelo codificador de entropia de forma a se produzir o *bitstream* comprimido.

A reconstrução do fluxo de dados dá-se da seguinte forma

1. Cada macrobloco quantificado *X* é reescalonado e passado pela IDCT de forma a produzir a imagem residual descodificada D'. É de lembrar que o processo de quantificação é um processo com perdas pelo que a imagem residual D' vai ser aproximação da imagem inicial D, com alguma distorção associada.
2. A predição *P* é adicionada à imagem residual D' de forma a se obter o macrobloco reconstruído. Após este passo cada macrobloco é armazenado de forma a produzir F_n'. Esta imagem pode ser aplicada em predições futuras de imagens (F_{n+1}).

Modo de funcionamento do descodificador:

1. O *bitstream* passa pelo descodificador de entropia, no qual são extraídos os vectores de movimento, coeficientes e cabeçalhos de cada macrobloco.
2. Reordenação e codificação *run-level*, de forma a se produzir um macrobloco *X*, quantificado e transformado.

3. *X* é reescalonado e inversamente transformado com o objectivo de obter a imagem residual descodificada *D'*.
4. O vector de movimento é usado para determinar a posição exacta do macrobloco 16 x 16 na cópia do descodificador da imagem referência F'_{n-1}. Esta região torna-se a predição compensada de movimento P.
5. P é adicionada a D' de forma a reconstruir o macrobloco. Os macroblocos reconstruídos são armazenados para produzir a imagem final descodificada F'_n. É necessário que o codificador e o descodificador usem a mesma imagem de referência F'_{n-1} para a realização da predição da compensação de movimento. Devido a este facto existe no codificador e no descodificador um caminho de descodificação.

Após o estudo das características de codificação e compressão de vídeo torna-se mais simples a compreensão da codificação de um vídeo em formato MPEG-4 AVC/H.264, como se poderá verificar no próximo capítulo onde se realiza a análise detalhada da norma.

Capitulo 4 – Norma MPEG-4 AVC/H.264

Neste capítulo começa-se por caracterizar a norma MPEG-4 AVC/H.264 ao nível temporal, quanto ao seu desenvolvimento. De seguida analisa-se a mesma quanto ao seu design estrutural, partido para uma caracterização mais detalhada das camadas de abstracção da rede e de codificação de vídeo.

4.1 – Desenvolvendo o MPEG-4 até aos Dias de Hoje

Criar, desenvolver e manter actualizada a norma ISO/IEC 14496 (‘MPEG-4’) [20] é da responsabilidade do *Moving Picture Experts Group* (MPEG), um grupo de estudo que desenvolveu as linhas base (*standards*) para a *International Organization for Standardization* (ISO). A emergente recomendação H.264, também conhecida por MPEG-4 Parte 10 – “*Coding of moving vídeo*”, é um esforço conjunto do grupo MPEG e do grupo *Coding Experts Group* (VCEG), um grupo de trabalho da União Internacional das Telecomunicações (ITU) (*International Telecommunications Union*) [21][22].

O grupo MPEG (Tabela 3) foi o responsável pelos muito conhecidos padrões de codificação de vídeo e de áudio, MPEG-1 e MPEG-2, muito utilizados na comunicação e armazenamento de vídeo digital, é também responsável pelo padrão MPEG-7, conhecido como "*Multimedia Content Description Interface*", usado na descrição de objectos multimédia [23][24], e MPEG-21, norma responsável pela compatibilidade da comunicação de dados máquina de uma forma inequívoca e secura [24]. O grupo VCEG foi responsável pela primeira e amplamente usada vídeo telefonia (H.261) e pela sua sucessora H.263, tratou ainda do desenvolvimento, algo precoce, do projecto H.26L, também conhecido como MPEG-4 Parte 10. Os dois grupos criaram o *Joint Video Team* (JVT) de forma a finalizarem o projecto H.26L para poderem criar a norma internacional H.264/MPEG-4 Parte 10, esta posteriormente foi publicada tanto pelo ITU-T como pelo ISO/IEC.

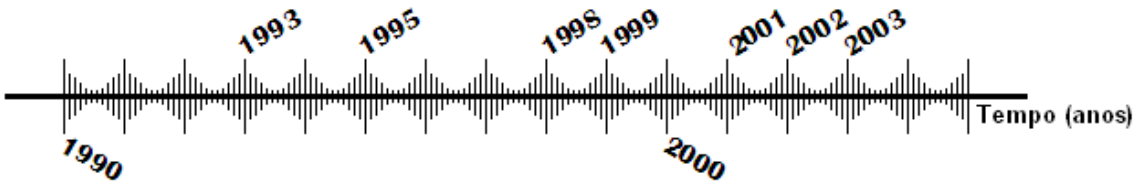

Figura 31- Comportamento no tempo do grupo MPEG-4 no desenvolvimento da norma H.264

No ano de 1993, foi lançado o projecto MPEG-4 e no mesmo ano foram lançados os primeiros resultados do estudo para o projecto H.263. No ano de 1995 o grupo “abriu as portas” (convidou) a propostas provenientes dos grupos de engenharia que se encontravam

em contacto com a tecnologia MPEG, neste momento impôs como referência a eficiência da codificação e de conteúdo de base bem como as funcionalidades inerentes a este tipo de projecto. Nessa data foi escolhido o H.263 como a tecnologia a utilizar como núcleo para o desenvolvimento de codificadores de vídeo.

Em 1998 o grupo MPEG/VCEG apresenta disponibilidade para a recepção de propostas para a norma H.26L, devido a este facto no ano seguinte (1999), dá-se a publicação na norma MPEG-4 Visual, inicialmente definida como H.26L. No ano 2000, o grupo lança o convite à comunidade para a apresentação de propostas de ferramentas de codificação de vídeo. Em 2001, o grupo MPEG lança a segunda versão da norma MPEG-4 *Visual*, no qual define o H.26L como proposta base para o desenvolvimento do MPEG-4 Parte 10, neste mesmo ano o grupo cria uma nova secção de trabalho a JVT – "*Joint Video Team*". Em 2002, são publicadas as primeiras revisões à norma MPEG-4 *Visual Edition 2 – Amendment 1* e *2* dedicadas aos perfis de estúdio e de envio de sequências de vídeo (*streaming*), nesta mesma data o desenvolvimento da norma H.264 ao nível técnico é "congelado" (definem-se todas as características de valor acrescentado da norma H.264).

Em 2003, é publicada a versão final da norma H.264/MPEG-4 Parte 10 – "*Advanced Video Coding*", na qual são descritos, com todo o detalhe, os pontos a ter em conta no desenvolvimento de um CoDec de vídeo para a norma H.264/MPEG-4. Desde esta data o MPEG-4 tem vindo a ser implementado em diferentes aplicações, tendo a sua rota traçada para a transmissão de vídeo digital de alta definição em regime de *broadcast*, com forte possibilidade de vir a ser a tecnologia escolhida para a substituição da rede de televisão terrestre analógica.

Tabela 3 - Secções e responsabilidades do grupo MPEG (adaptada de [5][25])

Subgrupo	Responsabilidades
"*Requirements Systems*"	Identificar necessidades da indústria e as necessidades de novas normas.
Sistemas	Combinação de áudio, vídeo e informações relacionadas; transportar e combinar dados sobre mecanismos de entrega.
Descrição	Declarar e descrever itens de média digital.
Vídeo	Codificação de imagens em movimento
Áudio	Codificação de áudio
"*Synthetic Natural Hybrid Coding*"	Codificação de áudio e vídeo sintético com áudio e vídeo natural
Integração	Testes de adaptabilidade e de referência para software
Teste	Métodos de avaliação da qualidade subjectiva.
Implementação	Guias experimentais, estudos de viabilidade, implementação e orientações.
Ligação (*Liaison*)	Relações com outros grupos e organismos relevantes.
"*Joint Video Team*"	Normas MPEG

Com o processo de normalização o grupo MPEG, tem tido como objectivo a definição de características físicas para o descodificador, impondo de certa forma restrições na sintaxe e fluxo de bits (*bitstream*) e definindo os elementos de sintaxe do processo de descodificação, com este trabalho o grupo tem vindo a definir a forma como descodificadores de vídeo construídos segundo a norma produzam uma saída em conformidade, muito semelhante ao mesmo *bitstream* codificado de acordo com as restrições da norma. Devido ao facto do grupo MPEG (e a norma) se ter vindo a concentrar somente no descodificador (Figura 32), criou-se uma grande liberdade para a implementação do codificador de vídeo. Este facto cria uma grande incerteza quanto ao funcionamento do descodificador genérico, não existindo qualquer tipo de garantias quanto à correcta descodificação/reprodução da sequência codificada, bem como da qualidade da sequência descodificada [5][26].

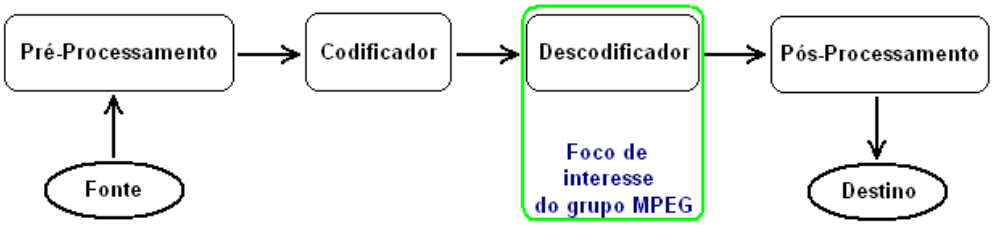

Figura 32 - Secção de interesse do grupo MPEG no desenvolvimento da norma H264/MPEG-4

4.2 – Aplicação e Destaques de Projecto (*Design* Estrutural)

A norma H.264 foi desenhada principalmente para criar soluções de carácter técnico nas seguintes áreas:

- Transmissão via cabo, satélite, modem de rede, DSL, terrestre.
- Transmissão em tempo real ou armazenamento em dispositivos magnéticos ou ópticos.
- Serviços de Conversação sobre ISDN (*Integrated Services Digital Network*), Ethernet, LAN, DSL, Wireless e redes móveis.
- "*Video-on-demand*" e serviços de multimédia sobre ISDN, modems de cabo, DSL, LAN e redes wireless.
- Serviços de mensagens multimédia (MMS – "*Multimedia messaging services*") sobre ISDN, DSL, Ethernet, LAN, wireless e redes móveis.

Com a norma H.264 existe ainda muito espaço á liberdade de criação, novas aplicações podem ser desenvolvidas e destacarem-se em novas redes com este tipo de tecnologia.

Face à possibilidade de flexibilidade e personalização os *designers* da norma H.264 criaram uma pilha protocolar de forma a se conseguir compatibilidade entre todas as aplicações.

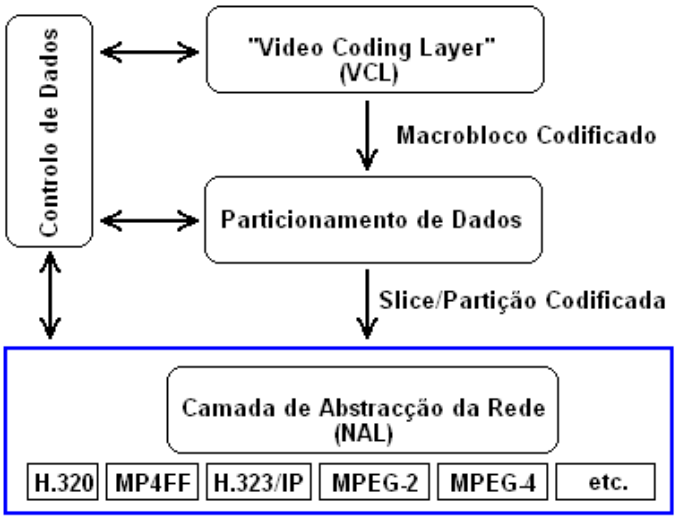

Figura 33 - Estrutura Interna (Pilha Protocolar) do Codificador H.264 (adaptada de [26])

A pilha (Figura 33) contém uma camada de codificação de vídeo ("*Video Coding layer*"-VCL), projectada para representar eficientemente o conteúdo de vídeo, uma camada de abstracção de rede ("*Network Abstraction Layer*"-NAL), que trata da formatação da representação VCL do vídeo e providencia um cabeçalho de informação para conveniência da variedade das camadas de transporte ou de armazenamento.

Relativamente aos métodos de codificação anteriores (por exemplo vídeo MPEG-2) o projecto MPEG-4 destacou algumas características que permitem reforçar a eficiência da codificação, como é o caso dos melhoramentos realizados ao nível da predição de imagens ao serem codificadas.

- **Blocos de dimensão variável, compensação de movimento com blocos de dimensão reduzida**, estas características permitem uma maior flexibilidade na selecção do tamanho de bloco (4 x 4 é a dimensão mínima para compensação de movimento na componente luminância) e da sua forma para a compensação de movimento, do que antigas codificações.
- **Compensações de movimento com um quarto de amostra**, antigas versões possuíam apenas meia amostra de precisão.
- **Vectores de movimento sobre os limites da figura (técnica da extrapolação da fronteira)**, enquanto na versão MPEG-2 e seus precedentes apenas existia cobertura nas regiões previamente descodificadas da imagem referência, esta característica foi herdada pelo H.264 e é proveniente do seu antecessor H.263.
- **Várias imagens referência para a compensação de movimento**, imagens preditas e codificadas (chamadas de *P*), no MPEG-2 e seus precedentes utilizava-se apenas a imagem anterior para realizar a predição da imagem.
- **Dissociação da imagem, métodos de representação para a capacidade de referênciamento de imagens**, em normas mais antigas *frames* codificadas usando métodos de codificação Bi-preditivos não poderiam ser usadas como imagens referência na predição de outras imagens de vídeo. Ao se eliminar esta restrição a nova norma prevê uma maior flexibilidade do codificador, e em alguns casos usar as imagens como referência para uma melhor aproximação da imagem a ser codificada.

- **Predição ponderada**, uma inovação no H.264, permite que sejam atribuídos "pesos" diferentes ao mesmo sinal a realizar predição no codificador. Este novo campo pode melhorar de forma muito significativa a eficiência da codificação de cenas que contenham desvanecimentos de imagem, pode ainda ser usado de forma flexível para outro tipo de aplicações.
- **Predição espacial direccional para codificação Intra**, uma nova técnica de extrapolar os limites de partes anteriormente descodificadas da imagem corrente. Partes da imagem corrente são aplicadas a regiões de imagem que são codificadas como Intra (codificadas sem referência de conteúdo de outras imagens), este processo melhora a qualidade da predição realizada e faculta a predição de áreas vizinhas que não estão codificadas usando codificação Intra.
- **Filtragem e descompactação de blocos em *loop***, a codificação de blocos de vídeo produz artefactos conhecidos como artefactos de bloqueio ("*blocking artifacts*"), estes podem originar tanto a predição como a diferença residual de codificação entre fases, no processo de descodificação [26].

Não só os métodos de predição de imagens foram melhorados na evolução para H.264, outras partes do modelo também foram melhoradas como forma de melhorar a codificação.

- **Tamanho do bloco da transformada de dimensões reduzidas**, em contrapartida às versões antigas que usavam tamanhos de bloco de 8 x 8 o H.264 foi desenhado para usar blocos de dimensão 4 x 4. Este tamanho permite ao codificador representar sinais de uma forma localmente mais adaptada, esta característica reduz a quantidade de artefactos que apareciam com formas diferentes de operar.
- **Blocos das transformadas processados de forma hierárquica**, pelo contrário, à maioria dos casos, blocos de dimensão 4 x 4 são uma vantagem, existem casos em que este tamanho não é vantajoso, casos em que os sinais contêm demasiada correlação entre eles. O H.264 permite, utilizando uma hierarquização de transformadas alargar, para as baixas frequências, o tamanho do bloco de crominância a 8 x 8. Para além do alargamento do tamanho do bloco de crominância, permite ainda ao codificador seleccionar uma codificação especial do tipo codificação Intra, permitindo assim uma prorrogação do período de Luma, de forma a transformar as informações das baixas frequências num bloco de dimensão 16 x 16 de uma forma semelhante à utilizada no alargamento do bloco para a crominância.
- **Comprimento de palavra de dimensão reduzida**, todos os *designs* prévios tiveram a necessidade de recorrer a codificadores e descodificadores complexos, mais lentos e com maior quantidade de dados a processar. Em contrapartida o processamento a realizar no H.264 é relativamente simples e requer apenas 16 bits.
- **Correspondência perfeita na transformada inversa**, o H.264 conseguiu uma correspondência quase exacta na passagem pelo codificador e depois pelo descodificador, diminui a quantidade de erros significativamente.
- **Codificador de entropia aritmético**, um avançado codificador de entropia incluído no H.264. Este foi utilizado no H.263 como carácter opcional, no H.264 viu-se necessário usar esta técnica na sua forma mais avançada, criando-

se assim um robusto e muito poderoso codificador de entropia, o CABAC – "*Context-adptative binary arithmetic coding*"

- **Contexto adaptativo do codificador de entropia**, os dois métodos usados no H.264, o CAVLC – "*Context-adptative variable-lenght coding*" e o CABAC, ambos usam "*context-based adaptivity*" para melhorar o desempenho relativamente a normas anteriores [26].

A Robustez a perdas de dados, a erros e a flexibilidade em operar sobre uma variedade de ambientes de rede foi tida em conta num conjunto de características introduzidas no projecto da norma H.264,

- **Estrutura de parâmetros**, a organização estrutural de todos os parâmetros produz informação detalhada e precisa dos conteúdos a serem transmitidos, este facto permite que quando existe a perda de pacotes na rede, estes sejam repostos sem perdas de pacotes importantes para visualização da sequência de vídeo.
- **NAL – Unidade de sintaxe estrutural**, cada estrutura de sintaxe, no H.264 é colocada dentro de um pacote lógico chamado de unidade NAL (NALU – *NAL unit*). Mais tarde quando se dá a transmissão de um *bitstream* a NALU, permite uma maior personalização do método de execução do conteúdo do vídeo adequado à rede onde se encontra a ser transmitido.
- **Tamanho de *slice* flexível**, em contrapartida ao tamanho fixo de *slice* do MPEG-2, que reduz a eficiência da transmissão de dados, aumenta a quantidade de cabeçalhos a circular na rede, o tamanho das *slices* no H.264 é altamente flexível, como era o caso do MPEG-1.
- **Ordenação de macroblocos flexível (FMO – "*Flexible Macroblock Ordering*")**, permite partir a imagem em regiões chamada de *slice groups* à medida que vão sendo produzidas, cada grupo torna-se um subconjunto independente, passível de descodificar. Quando utilizado de forma eficaz produz uma reordenação flexível dos macroblocos o que permite melhorar eficazmente a robustez dos dados a circular na rede. Com blocos de dimensão menor não existe tanto o problema de perdas devido a escassez de espaço para envio do pacote.
- **Ordenação arbitrária de *slice* (ASO – "*Arbitrary slice ordering*")**, dado que cada *slice* de uma imagem codificada pode ser (aproximadamente) descodificada independentemente das outras *slices* da imagem, a equipa de projecto do H.264 permitiu o envio e a recepção de *slices* de imagem numa ordem diferente que a ordem de sequência de vídeo. Esta capacidade, usada já anteriormente na norma H.263, melhora o funcionamento em tempo real dado que diminui os tempos atraso, principalmente em aplicações de tempo real com redes que permitem a entrega de pacotes não ordenada.
- **Imagens de redundância**, a fim de melhorar a robustez quanto à perda de dados, a equipa de projecto para o H.264 introduziu no codificador a capacidade de enviar representações redundantes de uma região da imagem, permitindo uma representação da imagem com menor quantidade de erros, este facto permite anular possíveis perdas de pacotes na rede de transmissão.
- **Particionamento dos dados**, dado que algumas informações para representação codificada de uma região são mais importantes ou mais valiosas

que a própria representação do conteúdo da sequência de vídeo, o H.264 permite a separação da *slice* em três partições diferentes para a transmissão na rede, dependendo apenas da divisão em prioridades dos elementos de sintaxe.

- **Sincronização e comutação de imagem (SP/SI – "*Synchronization/Switching Pictures*")**, o projecto H.264 inclui um recurso que permite realizar o exacto sincronismo do processo de descodificação via imagens de tipos diferentes (envio de imagens I no *stream* de vídeo). Este tipo de recurso permite que durante o processo de descodificação se possa comutar entre diversos descodificadores sem que o utilizador note perda de eficiência (Qualidade de Serviço), permite também algumas vantagens para o utilizador, dado que com este tipo de recurso é possível ter sistemas de *fast-forward* e *fast-reverse* [26].

4.3 – Camada de Abstracção de Rede (NAL)

A NAL foi criada de forma a tornar transparente a camada de rede, para permitir de forma simples e eficaz o uso do VCL para uma variedade de sistemas. A NAL permite mapear os dados do H.264 VCL numa variedade de camadas de transporte de rede, tais como:

- RTP/IP ("*Internet Real-Time Transport Protocol*"), para qualquer tipo de aplicação em tempo real com ou sem fios, para serviços de Internet de voz ou de *stream* de vídeo.
- H.32X para ligações com e sem fios para serviços de conversação.
- MPEG-2 sistemas de *broadcast.*

A possibilidade de personalização de conteúdos de vídeo fica um pouco fora do âmbito da norma, as necessidades de personalização para cada aplicação específica permitem criar diversos ambientes cumprindo sempre a norma, devido à especificidade de cada aplicação, no âmbito da normalização o grupo H.264, antecipou no *design* da NAL uma variedade de configurações de mapeamento, é claro que é difícil inserir nos sistemas todas as configurações possíveis e imaginárias, mas este esforço permitiu uma maior compatibilidade de sistemas logo à partida. Alguns dos modos de mapeamento incidem sobre alguns conceitos chave das unidades NAL (NALU), como é o caso do *Bitstream*, o formato dos pacotes NAL, parâmetros de ajuste e mesmo acesso às unidades. Alguns dos conceitos chave da NAL são explicados abaixo:

A. Unidades NAL (NALU)

Os dados codificados de um vídeo são organizados em Unidades NAL (NALU – "*NAL units*"), cada uma corresponde a um pacote que contém um número inteiro de bytes. O primeiro, é um cabeçalho que contém a indicação do tipo de dados que a NAL possui, os restantes bytes contêm a carga útil (*payload*) do tipo indicado pelo cabeçalho. A carga útil da NAL é intercalada conforme o necessário com emulação de prevenção de bytes (*emulation prevention bytes*), que são bytes inseridos com um valor específico de forma a

prevenir um dado padrão de dados, chamado de *start code prefix*, de ser acidentalmente gerado dentro da carga útil dos dados a enviar.

A estrutura da unidade NAL, especifica um formato genérico para a utilização em sistemas de transporte orientados ao pacote ou ao *Bitstream*. Uma sequência de unidades NAL é chamada de *Stream* de unidades de NAL (*NAL Unit Stream*).

B. Unidades NAL em uso num formato *Byte-stream*

Alguns sistemas, como o caso do H.320 e o MPEG-2/H.222.0 exigiam, a entrega total ou parcial do *stream* NAL como um fluxo ordenado de bits ou bytes dentro dos quais os limites das unidades NAL deviam ser identificáveis a partir de padrões codificados com os dados.

Para o funcionamento em tais sistemas, a especificação H.264 define o formato *byte stream*. Neste formato a unidade NAL contém um prefixo identificador específico, composto por três bytes, chamado de prefixo de início de código (*start code prefix*). O uso de emulação de prevenção de bytes garante que cada prefixo de início de código da NALU é único e identifica o início de cada NAL nova.

C. Unidades NAL no Sistema de transporte por pacotes

Em outros sistemas (por exemplo, protocolo de Internet ou sistemas com RTP), os dados codificados são transportados em pacotes, pacotes estes que são formados pelo protocolo de transporte. O perímetro de identificação do perímetro das NALUs pode ser estabelecido sem utilização de prefixos identificadores de início de NALU, nestes sistemas, a inclusão de prefixos nos dados seria um desperdício da capacidade de carga da rede. Assim as NALUs são transportadas em pacotes de dados sem necessidade de recurso a prefixos de início de NALU.

D. NALUs do tipo VCL e Não VCL

As NALUs são classificadas em VCL e não VCL (non-VCL). As NALUs VCL contêm dados que representam valores das amostras das imagens de vídeo, em contrapartida, as NALUs não VCL contêm qualquer tipo de informação adicional não essencial à representação da imagem de vídeo, tais como, parâmetros de configuração, cabeçalhos de dados importantes que podem ser aplicados a um elevado número de unidades NAL VCL, e informações complementares, como é o caso das informações de tempo (*timing*) e outros dados suplementares que possam aumentar a fiabilidade do descodificador do sinal de vídeo.

E. Configuração de Parâmetros

Dá-se nome de parâmetro a uma variável que raramente mude e que ofereça à descodificação um grande número de facilidades no processo de interpretação. Existem dois tipos de parâmetro:

- **Sequência**, que se aplica a uma série consecutiva de imagens codificadas de vídeo, chamadas de sequência de vídeo codificada.
- **Imagem**, que se aplica ao processo de descodificação de uma ou mais imagens individuais dentro de uma ou mais sequências de vídeo codificadas.

Cada NALU VCL contém um identificador, que se refere ao conteúdo do parâmetro da imagem relevante, desta maneira, uma pequena quantidade de informação (o identificador), consegue referir-se a uma grande quantidade de dados (de informação, os parâmetros de configuração) sem ter de repetir esta quantidade de informação em cada NALU.

F. Unidades de Acesso (*Access Units*)

Um conjunto de NALUs em uma determinada forma é referido como um acesso à unidade. A descodificação de cada acesso a uma unidade resulta numa imagem descodificada. Cada acesso à unidade contém um conjunto de unidades VCL NAL que em conjunto compõem a imagem primária codificada (*primary coded picture*). Também pode ser pré-fixado com o início de cada unidade um delimitador de acesso (*unit delimiter*), que simplifica o processo de localização do início da unidade de acesso. Algumas informações complementares (*supplemental enhancement information*) contendo dados da imagem, como informações de tempo (*timing*) também podem preceder a imagem primária codificada.

A imagem principal codificada consiste num conjunto NALUs VCL, consistindo em fatias (*slices*) ou fatias de partições de dados (*slices data partitions*) que representam as amostras das imagens de vídeo. Após a primeira imagem codificada algumas NALUs adicionais que contêm informações de áreas redundantes da mesma imagem de vídeo podem ser também codificadas, estas são referidas como imagens redundantes codificadas, e estão disponíveis para a utilização por um descodificador na recuperação de dados corrompidos e perdidos nas primeiras imagens codificadas. No final quando toda a imagem se encontra descodificada a última NALU possui um identificador de fim de sequência de vídeo, de forma a informar o descodificador que a sequência de vídeo termina nessa NALU.

G. Sequências de vídeo codificadas

Uma sequência de vídeo codificada consiste numa série de unidades de acesso que são sequências num *stream* de NALUs e que usa apenas um parâmetro de configuração. Cada sequência de vídeo pode ser codificada de forma independente de uma outra dando apenas o parâmetro correcto para tal,

que pode ser transportado *In-band* ou *out-band*. No início de uma sequência de vídeo codificada existe sempre uma unidade de acesso do tipo *instantaneous decoding refresh* (IDR). Uma unidade de acesso IDR contém a informação Intra da imagem – uma imagem que pode ser descodificada sem qualquer tipo de imagem precedente. A presença de uma unidade IDR indica que nenhuma imagem subsequente à imagem Intra no *stream* requererá de imagem referência Intra, a fim de ser descodificada.

Um *stream* NALU pode conter uma ou mais sequências de vídeo codificadas.

4.4 – Camada de Codificação de Vídeo ("*Video Coding layer*"-VCL)

Tal como em todas as normas de vídeo anteriores da ITU-T e da ISO/IEC JTC1, desde o H.261, o VCL segue uma arquitectura de tipo híbrido da abordagem *block-based vídeo coding* em que cada imagem é representada por blocos em unidades de luminância e crominância, chamados de macroblocos. O algoritmo usado é um do tipo híbrido de predição de imagem Inter, de forma a explorar as dependências temporais estatísticas e transformar a codificação de previsão residual para a exploração das dependências espaciais estatísticas. O VCL não fornece qualquer tipo de elemento de codificação, componente que mesmo assim é o componente que fornece a maioria das melhorias mais significativas ao nível da eficiência na compressão em relação às normas anteriores de codificação de vídeo.

A. Imagens, *Frames* e Campos

Uma imagem codificada em H.264 consiste numa sequência de imagens codificadas. Uma imagem codificada pela norma pode ser representada tanto por uma frame completa como por um único campo, como é o caso do MPEG-2 [20].

De forma geral, uma frame de vídeo pode ser considerada como tendo dois campos entrelaçados, um superior e um inferior. Se os dois campos forrem capturados em dois instantes de tempo diferentes, a frame é apelidada de uma frame do tipo entrelaçado (*interlaced*), caso contrário é chamada de frame progressiva (*progressive*) [5].

B. Espaço de Cor YC_bC_r e o formato 4:2:0

Como se sabe o olho humano só se apercebe de uma cena em conteúdo em proporções diferentes em relação à cor e ao brilho, (explicado no capítulo 2) como se sabe o olho responde melhor a impulsos de brilho, as transmissões de vídeo tiram proveito dessa característica, a televisão analógica usufrui desta característica à muito tempo. Em normas como o H.264 bem como em normas anteriores, isto é realizado, recorrendo ao espaço de cor YC_bC_r, juntamente com a redução da resolução da amostragem das componentes de crominância C_b e C_r. O espaço de cor usado pelo H.264 separa a representação da cor em

três componentes chamadas de Y, C_b e C_r, a que Y corresponde à luminância, C_b à crominância azul e C_r à crominância vermelha.

Porque o olho humano reage de forma diferente à luminância e à crominância, o H264 usa uma estrutura representativa em que a componente luminância é quatro vezes maior que a componente de crominância, como se sabe este comportamento corresponde ao formato de amostragem 4:2:0, este apresenta uma precisão de 8 bits por amostra [5].

C. Divisão da imagem em Macroblocos

A imagem é particionada em blocos de tamanho fixo de 16 x 16 para a luminância e 8 x 8 para cada componente de crominância. A divisão em Macroblocos foi adoptada em recomendações do ITU-T e da ISO/IEC JTC1 desde a norma H.261. Os macroblocos são o padrão para o qual o processo de descodificação é especificado [26].

D. *Slices* e Grupos de *Slices*

Slices são uma sequência de macroblocos ordenados por *raster scan* quando não se está a usar FMO. A figura pode ser ser dividida em uma ou mais fatias (*slices*) (Figura 34) Uma imagem é pois uma colecção de uma ou mais *slices* na norma H.264, as mesmas são auto-contidas, no sentido em que a sequência é dada como activa, isto quer dizer que uma *slice* pode ser descodificada sem o recurso a qualquer informação proveniente de qualquer outro lado.

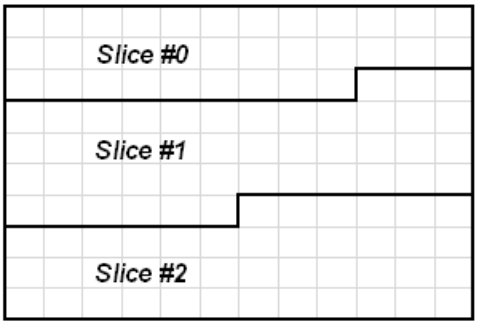

Figura 34 - Divisão de uma imagem em *slices*, sem usar FMO (adaptada de [26])

Com a possibilidade de uso de FMO, o método de divisão em *slices* torna-se um pouco diferente, com FMO a imagem pode dividir-se em muitas "*slices*" (*group slices*), tantas quantas o número máximo de *slices* entrelaçadas possíveis (Figura 35) [26].

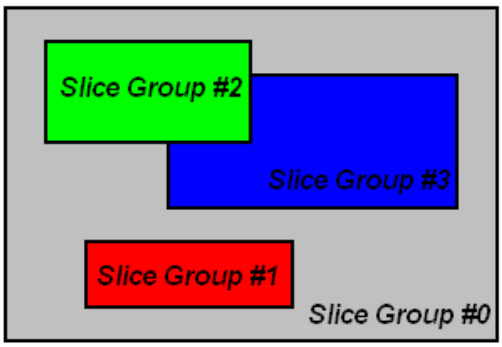

Figura 35 - Subdivisão da *frame* em *slices*, usando FMO (adaptada de [26])

Usando ou não FMO, cada *slice* pode ser codificada da seguinte forma:

- ***Slice* I**, uma *slice* em que todos os macroblocos da imagem são codificados usando codificação com predição do tipo Intra (ver quantidade de energia da *slice* I na Figura 36).
- ***Slice* P**, em adição às formas de codificar as *slices* do tipo I, alguns macroblocos da *slice* P são codificados usando predição do tipo Inter em no máximo por cada macrobloco existe um vector com um sinal com predição compensada de movimento por cada bloco de predição (ver quantidade de energia da *slice* P na Figura 37).
- ***Slice B***, derivadas das *slices* P, alguns macroblocos também podem ser codificados usando predição Inter com, no máximo, dois vectores com predição compensados de movimento, por bloco de predição.
- ***Slice SP,*** chamada de comutação de P, facilita a troca (switching) entre *streams* codificadas, contem macroblocos P e I.
- ***Slice SI***, também chamada de comutação de I, permite uma correspondência exacta de um macrobloco numa *slice* SP para acesso aleatório e recuperação de erros.

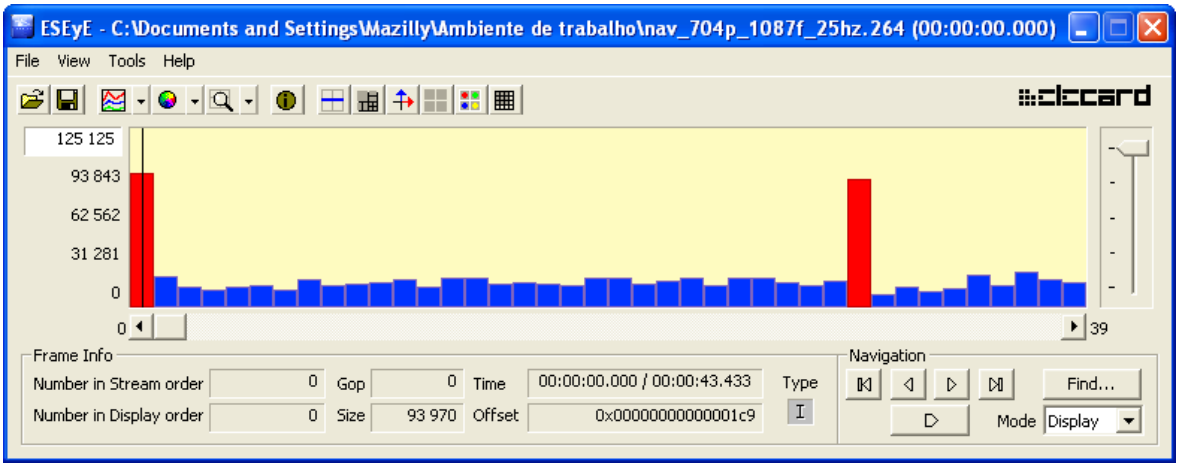

Figura 36 - *Slice* I (Análise via Elecard [27] de *stream* de vídeo)

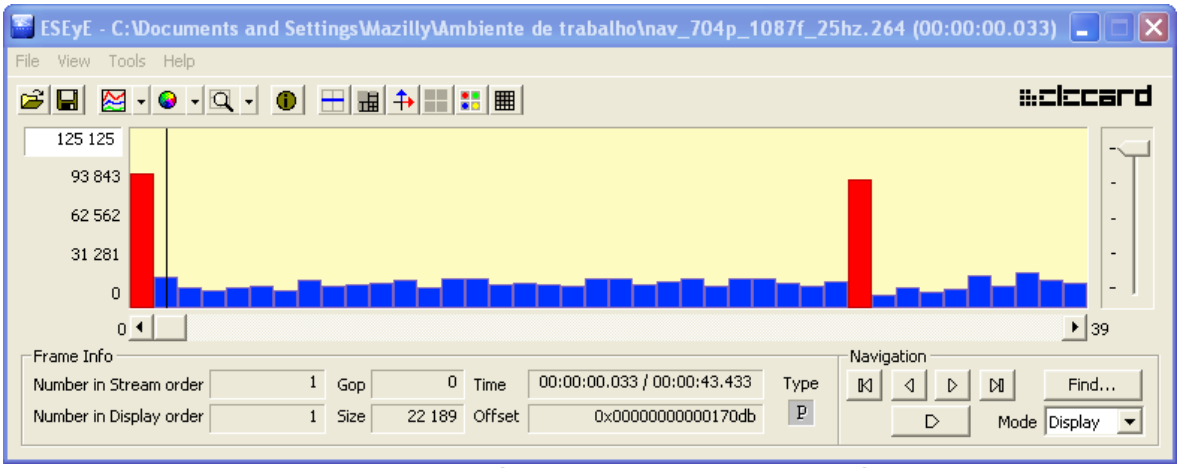

Figura 37 - *Slice* P (Análise via Elecard [27] de *stream* de vídeo)

E. Processos de Codificação e Descodificação para macroblocos

Todas as amostras de luminância e crominância de um macrobloco são espacialmente e temporalmente preditas e o resultado residual da predição é codificado usando uma transformada de codificação. Para propósitos da codificação da transformada, cada componente de cor é subdividida em pequenos blocos de dimensão 4 x 4. Cada bloco é transformado, usando uma transformada inteira, e os coeficientes resultantes dessa transformação são quantificados e codificados usando codificadores de entropia [26].

F. Codificação Adaptativa de Imagem/Campo (*Frame/Field*)

Com a utilização de vídeo entrelaçado, sabe-se que duas linhas adjacentes têm uma dependência estatística reduzida, quando comparadas com as do vídeo progressivo, isto significa, que é mais eficiente codificar separadamente cada campo de vídeo entrelaçado. A norma H.264 permite que o codificador decida qual o procedimento a adoptar para codificar o campo (*field*):

- Combinando o campo par e o campo impar juntos, codificando-os como uma única imagem de um vídeo progressivo (**modo imagem**).
- Codificando o campo par e ímpar juntos comprimindo-os numa única imagem, mas no acto da codificação da imagem, separar os macroblocos pares de dois macroblocos verticalmente adjacentes em pares de macroblocos de dois campos.
- Codificando separadamente os campos par e impar (**modo campo**).

A opção de escolha entre os diferentes modos de codificação dos campos, pode ser realizada de modo adaptativo para cada campo da sequência. A este procedimento dá-se o nome de *Picture-Adaptive Frame/Field* (PAFF), quando a escolha incide nos **modos de imagem** e de **campo.**

Quando uma imagem é codificada em dois campos, então cada um dos mesmos é dividido em Macroblocos e é codificado como se de uma imagem completa se trata-se, excepto se:

- A compensação de movimento utilizar como referência campos em vez de imagens.
- O varrimento em ziguezague dos coeficientes da transformada for diferente.
- Não se realizar filtragem horizontal de macroblocos dos campos.

Quando numa *frame* existe uma mistura de regiões, onde se tem movimento em algumas zonas, é errado pensar-se que o comportamento do codificador vai ser semelhante ao caso de uma *frame* ausente de movimento, sabe-se que a codificação de imagens estáticas se dá de um modo mais eficiente. No decorrer do desenvolvimento da norma H.264 o modo de codificação PAFF mostrou uma redução de débito de 16% a 20% em relação à codificação de uma imagem estática [26].

Ao processo de escolha entre os diversos modos de codificação, campo ou imagem na mesma cena chama-se *Macroblock-Adaptive Frame/Field* (MBAFF) [26].

G. Predição Intra – *Intra Frame Prediction*

- Intra 4 x 4

O modo Intra 4 x 4 (ou I_4x4) baseia-se no resultado da predição realizada a cada bloco de luminância de dimensão 4 x 4, este encontra-se bem adaptado à codificação de imagem com elevado detalhe. Cada bloco 4 x 4 é predito das amostras espaciais vizinhas (Figura 38). As 16 amostras presentes num bloco de dimensão 4 x 4, de a a p, são resultados da predição realizada aos blocos vizinhos, de A a M.

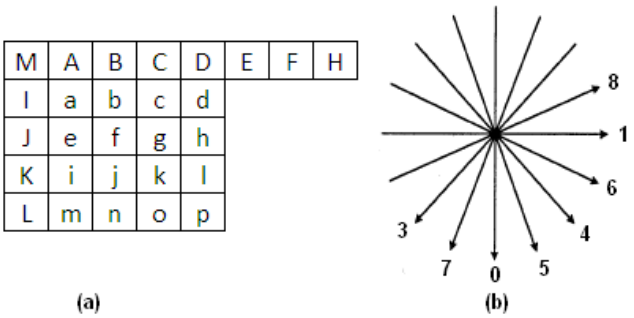

Figura 38 - Predição no modo I_4x4 (a) e Direcções de predição para o modo I_4x4 (b) (adaptada de [5][20])

No modo Intra 4 x 4 existem 9 modos de realizar predição direccional de estruturas na imagem (Figura 39) [26].

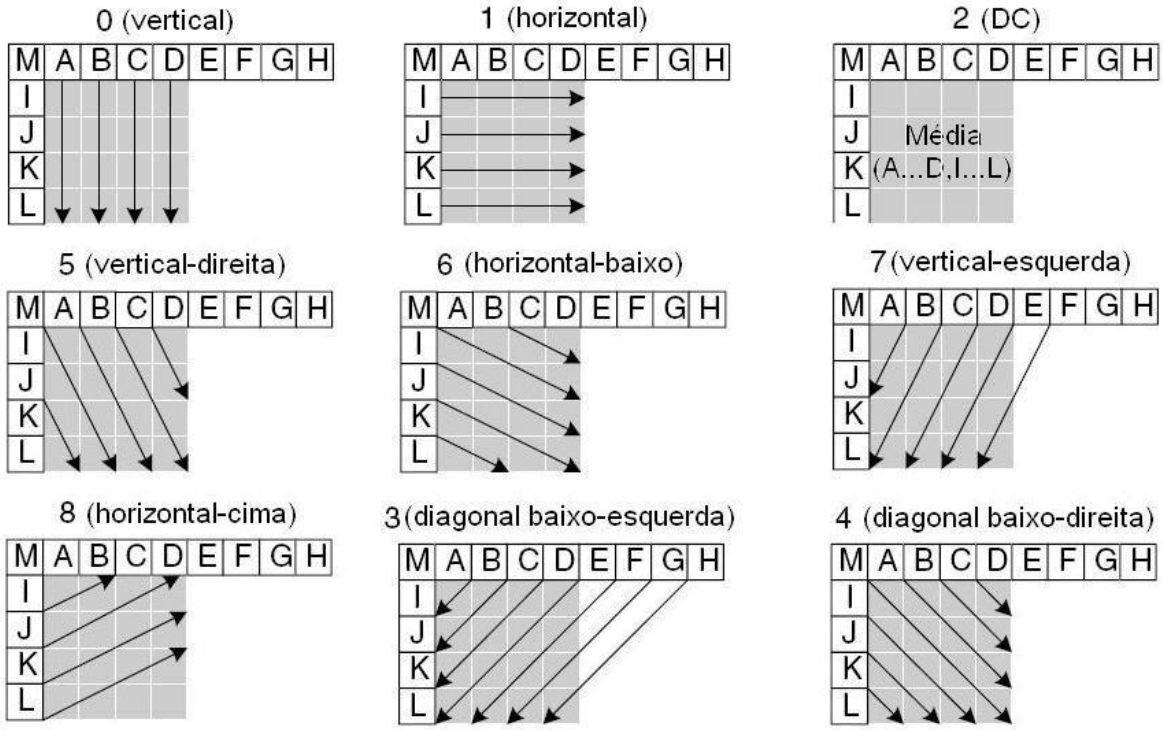

Figura 39 - 4 x 4 modos de predição de luminância (adaptada de [5])

- Intra 16 x 16

Este modo (Intra_16x16 ou I_16x16) realiza predição de blocos de luminância de dimensão 16 x 16 e encontra-se adaptado à codificação de áreas "suaves" da imagem. Para além da luminância existe o mesmo tratamento para a crominância. Neste sentido existem quatro modos de realizar predição (Figura 40).

O modo Intra é também usado para realizar predição Intra de crominância, utiliza os mesmos quatro modos de operar mas a dimensão do bloco é reduzida para 8 x 8, uma vez que o comportamento da crominância é semelhante ao longo de grandes áreas [26].

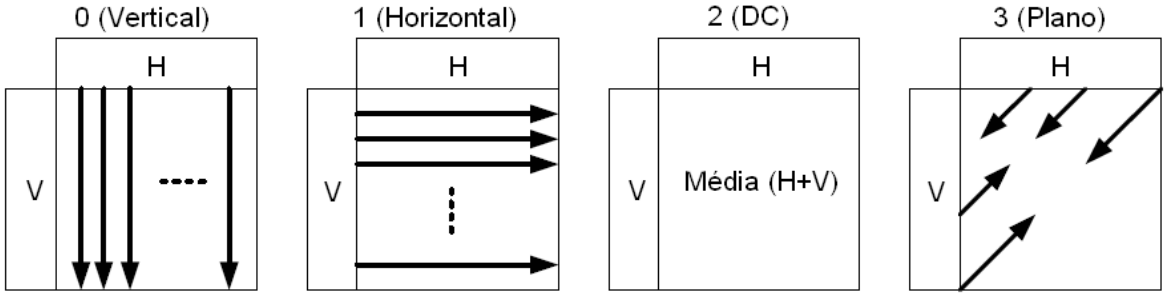

Figura 40 - Modos de Predição Intra_16x16 (adaptada de [5])

- Intra PCM

Para além dos dois tipos de predição mencionados anteriormente existe mais um o Intra PCM, também conhecido por I_PCM. Este tipo de codificação permite contornar os processos de predição e transformação, tratando do envio directo dos valores das amostras codificadas. O I_PCM serve para:

- ♦ Permitir que o codificador represente de forma precisa os valores das amostras.
- ♦ Fornecer uma forma de representação dos valores dos conteúdos anómalos da imagem sem aumentar a quantidade de dados a processar.
- ♦ Estabelecer um limite físico para o número de bits máximo que o descodificador deve processar para um macrobloco, sem degradar a eficiência de codificação.

Ao contrário de outras normas para a codificação de vídeo, que realizam predição Intra no domínio da transformada, no MPEG-4 AVC realiza-se predição Intra sempre no domínio espacial. De forma a evitar a propagação de erros é introduzido um modo Intra e identifica-se o modo como tal, para que a predição seja sempre realizada só entre macroblocos vizinhos codificados como Intra. Para além desta característica os modos de predição Intra não ultrapassam as fronteiras das *slices*, de forma a manter todas as *slices* independentes umas das outras [26].

H. Predição Inter – *Inter Frame Prediction*

- Compensação de movimento em árvore

A luminância de cada macrobloco pode ser dividida em quatro formas distintas (Figura 41):

- Uma partição do macrobloco 16 x 16.
- Duas partições de 8 x 16.
- Duas partições de 16 x 8.
- Quatro partições 8 x 8.

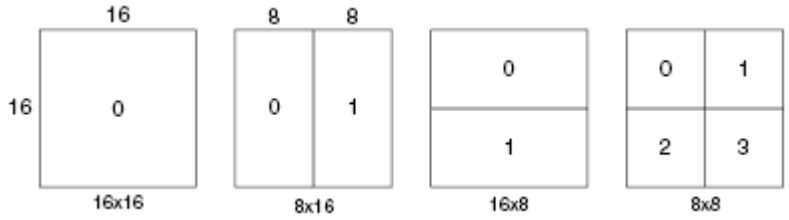

Figura 41 - Modos de Particionar Macroblocos (adaptada de [5] e [26])

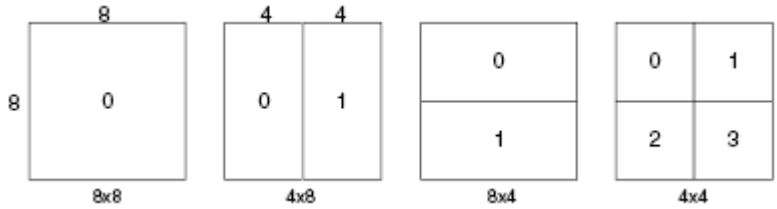

Figura 42 - Modos de particionar sub-macroblocos (adaptada de [5][26])

Se optar por escolher o modo 8 x 8 então cada um dos quatro sub-macroblocos do macrobloco original pode ser dividido em duas partições de 8 x 4, ou quatro partições de 4 x 4 (Figura 42). Estas partições aumentam o número de combinações possíveis por cada macrobloco. Ao acto de dividir macroblocos dá-se o nome de compensação de movimento em árvore (*tree structured motion compensation*). Devido a este facto torna-se necessário que o codificador envie a informação relativamente ao modo de como se encontra a dividir os macroblocos.

- Vectores de Movimento/Gerando amostras Interpoladas

Por cada sub-macrobloco com codificação Inter, a sua predição é realizada a partir de uma área igual na imagem referência. O desvio entre as duas áreas, o vector de movimento apresenta uma resolução de um quarto de amostra (*quarter-sample*) para a luminância e um oitavo de amostra (*eight-sample*) para a crominância. Como as amostras não existem é necessário recorrer a interpolação com base em amostras vizinhas (Figura 43) [26].

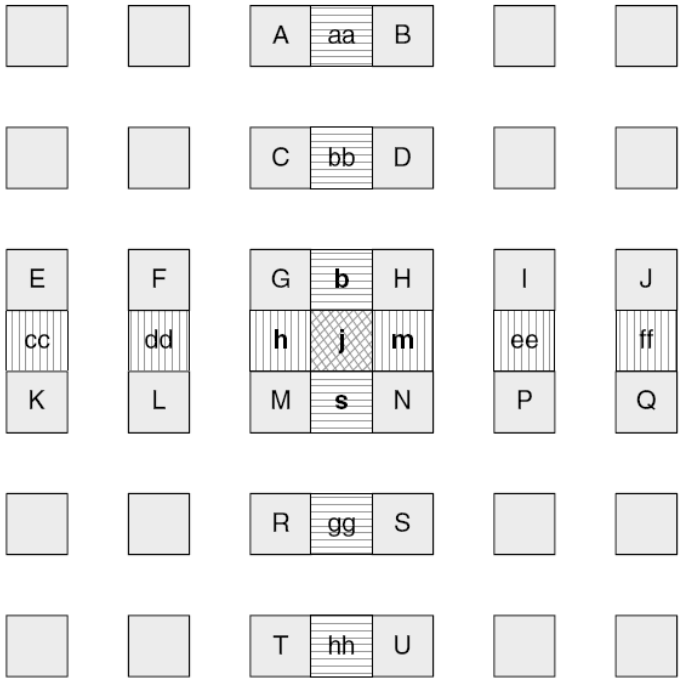

Figura 43 - - Interpolação para luminância *half-sample* (adaptada de [5])

As amostras com resolução *half-sample* são obtidas aplicando um filtro de uma dimensão de resposta finita (FIR - *Finite Impulse Response*) vertical e horizontalmente, tirando partido de 6 amostras horizontais e verticais [5][26]:

$$b = round\left(\frac{E - 5F + 20G + 20H - 5I + J}{32}\right) \qquad (4.1)$$

$$h = round\left(\frac{A - 5C + 20G + 20M - 5R + T}{32}\right) \qquad (4.2)$$

Onde b corresponde a uma amostra horizontal, calculada a partir das amostras E, F, G, H e I e h a uma amostra vertical, obtida via utilização das amostras A, E, G, M, R e T. O filtro FIR utilizado possui por cada amostra os pesos

$$\left[\frac{1}{32} \quad \frac{-5}{32} \quad \frac{5}{8} \quad \frac{5}{8} \quad \frac{-5}{32} \quad \frac{1}{32}\right]$$

(Nota: Os pesos do filtro FIR são os impostos pela norma [20])

As amostras *quarter-sample* são obtidas por interpolação linear (Figura 44), recorrendo às amostras *half-pel* vizinhas.

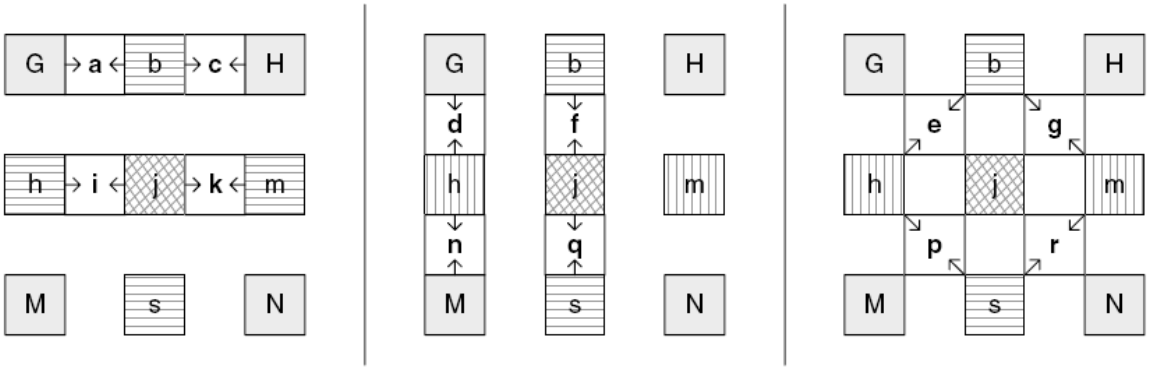

Figura 44 - Interpolação para luminância *quarter-sample* (adaptada de [5])

Por exemplo, se desejar determinar a amostra a (como poderia ser c, i, k, d, f, n e q), esta será determinada por interpolação linear do seguinte modo:

$$a = round\left(\frac{G + b}{2}\right) \qquad (4.3)$$

As outras amostras e, g, p e r, são determinadas utilizando amostras opostas na direcção diagonal, por exemplo a amostra **e** usaria as amostras b e h.

Os vectores *quarter-pel* de luminância, para serem determinados necessitam de uma resolução do tipo *eight-sample* na componente crominância, assumindo formatação da imagem do tipo 4:2:0. As amostras são geradas em intervalos de oito amostras em cada componente de crominância por interpolação linear. Cada amostra deste tipo é obtida por combinação linear das amostras inteiras vizinhas.

Na Figura 45 apresenta-se um exemplo da determinação a amostra de crominância em *eight-sample*.

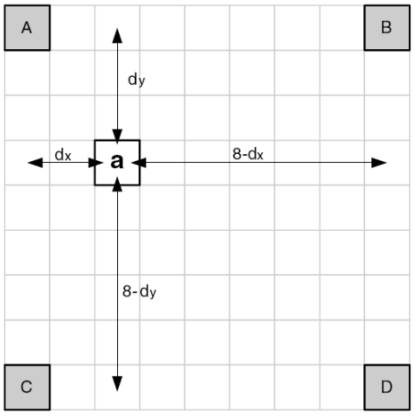

Figura 45 - Interpolação de componente de crominância *eight-sample* (adaptada de [5])

A amostra **a** é determinada, recorrendo às amostras vizinhas A, B, C e D, da seguinte forma:

$$a = round\left(\frac{(8-dx).(8-dy)A + dx.(8-dy)B + (8-dx).dyC + dx.dyD}{64}\right) \quad (4.4)$$

Como da Figura 45 se retira que dx é 2 e dy é de 3 "quadrículas". Então:

$$a = round\left(\frac{30A + 10B + 18C + 6D}{64}\right) \quad (4.5)$$

I. Transformada, Escalonamento e Quantificação

De modo semelhante a antigas normas, a H.264 utiliza codificação por transformada para a predição residual. No entanto, no H.264, a transformada é aplicada a blocos de dimensão 4 x 4, e em vez de uma transformada discreta dimensional 4 x 4 (DCT) é usada uma transformada inteira, com propriedades semelhantes às da DCT 4 x 4. A matriz **transformada** é:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \quad (4.6)$$

- *Como determinar a matriz H:*

Relembrando o estudo realizado no capítulo 3 em relação à transformada,

$$Y = AXA^T \tag{4.7}$$

Onde, $a = \frac{1}{2}$, $b = \sqrt{\frac{1}{2}}cos\left(\frac{\pi}{8}\right)$, $c = \sqrt{\frac{1}{2}}cos\left(\frac{3\pi}{8}\right)$ e

$$A = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & c \end{bmatrix} \tag{4.8}$$

Factorizando a equação 4.7 obtém-se

$$Y = (CXC^T) \otimes E \tag{4.9}$$

Onde Y é igual a, com $d = \frac{c}{b} \cong 0{,}414$

$$Y = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & d & -d & -1 \\ 1 & -1 & -1 & 1 \\ d & -1 & 1 & -d \end{bmatrix} \begin{bmatrix} \quad X \quad \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & d & -d & -1 \\ 1 & -1 & -1 & 1 \\ d & -1 & 1 & -d \end{bmatrix}^T \right) \otimes \begin{pmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{pmatrix} \tag{4.10}$$

A operação $(\mathrm{CXC^T})$ consiste no núcleo (*core*) da transformada 2D. Onde E é a matriz de escalonamento dos factores (matriz de factores de escala). Y é então uma transformada escalada, em que os seus factores são resultantes do pós-escalonamento dos elementos do *core* com a matriz E. O símbolo ⊗ indica que cada elemento do core é multiplicado pelo escalar presente na mesma posição da matriz E (por exemplo posição 1x1 do core multiplica com posição 1x1 de E).

Para simplificar a implementação do sistema, realizam-se as simplificações seguintes, para d=0,5:

$$a = \frac{1}{2} \quad b = \sqrt{\frac{2}{5}} \quad d = \frac{1}{2}$$

Desta forma obtém-se o resultado seguinte,

$$Y = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} \quad X \quad \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \right) \otimes \begin{pmatrix} \frac{a^2}{1} & \frac{ab}{2} & \frac{a^2}{1} & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \\ \frac{a^2}{1} & \frac{ab}{2} & \frac{a^2}{1} & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \end{pmatrix} \tag{4.11}$$

Daqui se retira que a matriz C é igual à matriz H

Dado que a transformada inversa é definida com operações entre inteiros bem determinados, problemas de má correspondência são evitados. O processo de codificação básica é muito semelhante a antigas normas. No codificador, o processo inclui uma transformada para a frente, pesquisa ziguezague, escalonamento, arredondamento e quantificação seguido por um codificador de entropia. No descodificador, dá-se o inverso das operações que se realizam no codificador, excepto o processo de arredondamento que não se realiza no descodificador.

Exemplo: *Cálculo da saída da transformada DCT de um bloco de amostras de dimensão 4 x 4 e da saída do mesmo bloco mas usando a matriz H*

O bloco seguinte (X) é um bloco de amostras retirado de uma imagem de vídeo:

	j=0	*1*	*2*	*3*
i=0	5	11	8	10
1	9	8	4	12
2	1	10	11	4
3	19	6	15	7

Pelo método de aplicação da transformada DCT temos:

$$Y_1 = AXA^T = \begin{bmatrix} 35{,}000 & -0{,}079 & -1{,}500 & 1.115 \\ -3{,}299 & -4{,}768 & 0{,}443 & -9{,}010 \\ 5{,}500 & 3{,}029 & 2{,}000 & 4{,}699 \\ -4{,}045 & -3{,}010 & -9{,}384 & -1{,}232 \end{bmatrix}$$

Para o método alternativo obtemos:

$$Y_2 = (CXC^T) \otimes E = \begin{bmatrix} 35{,}000 & -0{,}158 & -1{,}500 & 1.107 \\ -3{,}004 & -3{,}900 & 1{,}107 & -9{,}200 \\ 5{,}500 & 2{,}688 & 2{,}000 & 4{,}901 \\ -4{,}269 & -3{,}200 & -9{,}329 & -2{,}100 \end{bmatrix}$$

Se calcular-mos a diferença entre Y_1 e Y_2:

$$Diferença = Y_1 - Y_2 = \begin{bmatrix} 0 & 0{,}079 & 0 & 0{,}008 \\ -0{,}295 & -0{,}868 & -0{,}664 & 0{,}190 \\ 0 & 0{,}341 & 0 & -0{,}203 \\ 0{,}224 & 0{,}190 & -0{,}055 & 0{,}868 \end{bmatrix}$$

Como se pode verificar os resultados obtidos são diferentes, mas no contexto do H.264 esta diferença não é significativa, o comportamento da transformada é quase idêntico no que se trata à compressão e apresenta um número significativo de vantagens. O núcleo da transformada $(\mathrm{CXC^T})$ pode ser calculado realizando apenas operações de adição, subtracção e deslocamentos (*shifts*), necessitando apenas de um acumulador de 16 bits para realizar todas as operações aritméticas. A última parte da transformada, produto externo do núcleo com E, requer uma multiplicação por cada coeficiente, mas esta pode ser embutida no processo de quantificação, tornando o cálculo da transformada, um processo mais rápido.

A quantificação dá-se de uma forma simples,

$$Z_{ij} = round\left(\frac{Y_{ij}}{Q_{step}}\right) \tag{4.12}$$

Sendo que Y_{ij} corresponde aos coeficientes da transformada descrita anteriormente, Q_{step}o tamanho do quantificador e Z_{ij} os coeficientes quantificados.

Se assumirmos que o núcleo da transformada é $W = CXC^T$, então temos

$$Z_{ij} = round\left(W_{ij}\frac{PF}{Q_{step}}\right) \tag{4.13}$$

Onde PF assume valores de a^2, $\frac{ab}{2}$ e $\frac{b^2}{4}$,mediante a posição (i,j). De facto ainda se pode simplificar mais o processo, as operações aritméticas podem ser simplificadas, se considerarmos o facto $\frac{PF}{Q_{step}}$ poder ser implementado da seguinte forma,

$$\frac{MF}{2^{qbits}} = \frac{PF}{Q_{step}} \tag{4.14}$$

Onde, $qbits = 15 + floor\left(\frac{QP}{6}\right)$ e deste modo obtém-se:

$$\begin{gathered} \left|Z_{ij}\right| = \left(\left|W_{ij}\right|MF + f\right) \gg (qbits + 1) \\ sinal\left(Z_{ij}\right) = sinal\left(W_{ij}\right) \end{gathered} \tag{4.15}$$

Com $f = \frac{2^{qbits}}{3}$ para blocos Intra e $f = \frac{2^{qbits}}{6}$ para blocos Inter.

Com a simplificação de operações deixamos de ter de divisões entre coeficientes e passamos a necessitar apenas de *shift registers* binários. Ao nível das necessidades de hardware, com esta implementação a quantidade necessária é muito menor. [5][26]

J. Codificador de Entropia

O H.264 usa dois métodos de codificação de entropia o CAVLC e o CABAC.[20][22]

i. CAVLC

Os códigos de comprimento variável aplicados no CAVLC são do tipo *Exp-Colomb*. Estes têm a forma de $[M\ zeros][1][INFO]$, onde $INFO$ é um campo de M bits que contém a informação. O tamanho de cada código é de $2M + 1$ bits.

Os códigos são gerados segundo a expressão,

$$M = floor\left(log_2(número_{do\ código} + 1)\right)$$
$$INFO = número_{do\ código} + 1 - 2^M \quad (4.16)$$

Tabela 4 - *Exp-Colomb Codewords*

Número de Código	Palavra-Chave (*Codeword*)
0	1
1	010
2	011
3	00100
4	00101
5	00110
6	00111
7	0001000
8	0001001
…	…

A descodificação destes códigos é um processo simples e obedece a regras, tendo sempre em atenção que para um $número_{do\ código} = 0 \rightarrow INFO = M = 0$.

Regras de descodificação de códigos Exp-Colomb:

1. Ler Código de zeros em M seguido do valor 1;
2. Ler campo INFO com M bits;
3. $número_{do\ código} = 2^M + INFO - 1$.

ii. CABAC

A eficiência do codificador de entropia pode ser melhorada utilizando para tal o CABAC, este tipo de codificação é muito eficiente para símbolos com probabilidades superiores a 50%. O uso de códigos adaptativos permite a adaptação a estatísticas não estacionárias dos símbolos.

Em comparação com o CAVLC, o CABAC permite uma redução no débito de transmissão entre 9% e 14%, sendo que se obtém melhores ganhos quando se está a utilizar sinais de vídeo entrelaçados [28].

K. Filtro de Remoção de Artefactos – *Deblocking Filter*

Uma característica muito particular de codificadores baseados em blocos é a produção acidental de estruturas visíveis com o tamanho de blocos. Este facto deve-se ao processo de reconstrução dos blocos, as bordas dos mesmos são reconstruídas com menor precisão, que o interior do bloco [26]. Por este motivo o H.264 define um filtro adaptativo de remoção de artefactos, que reduz o aparecimento deste tipo de problemas. Este filtro funciona da seguinte maneira:

1. Dadas as amostras $p0$, $q0$, $p1$ e $q1$ (Figura 46);
2. O parâmetro de quantificação QP, que depende dos limiares $\propto (QP)$ e $\beta(QP)$, que determinam a filtragem das amostras;
3. A filtragem de $p0$ e $q0$ só se concretiza se:

i. $|p0 - q0| < \propto (QP)$

ii. $|p1 - p0| < \beta\ (QP)$

iii. $|q1 - p0| < \beta\ (QP)$

Sendo $\beta(QP)$consideravelmente menor que $\propto (QP)$.

Desta forma, $p1$ e $q1$ só são filtrados se a condição for satisfeita:

$$|p2 - p0| < \beta\ (QP) \quad \vee \quad |q2.q0| < \beta\ (QP)$$

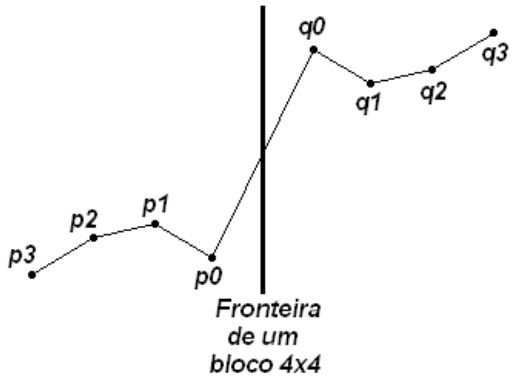

Figura 46 - Principio de Funcionamento do filtro de remoção de artefactos (adaptada de [26])

Com este filtro, os artefactos de blocos são reduzidos e as altas frequências da imagem permanecem inalteradas.

Figura 47 - *Frame* sem aplicação de filtro de remoção de artefactos (adaptada de [26])

Figura 48 - *Frame* com aplicação de filtro de remoção de artefactos (adaptada de [26])

Como se pode ver pela Figura 48, a imagem após passagem pelo filtro de remoção de artefactos fica com um aspecto mais "linear", menos rugosa do que a Figura 47. Como se pode ver na Figura 47 encontram-se assinalados pela caixa azul alguns artefactos de blocos, com a passagem pelo filtro de remoção dos mesmos, estes desaparecem, como se pode ver na mesma região mas na Figura 48.

Com esta análise detalhada da norma H.264 encontramo-nos prontos a enveredar por um caminho de análise do comportamento do vídeo ao nível do transporte e armazenamento da informação do vídeo, isto é o protocolo utilizado para o transporte dos pacotes de dados que contêm a informação da sequência de vídeo e formatos de vídeo utilizados para o armazenamento de sequências. Assim sendo o capítulo seguinte explicará o armazenamento e o transporte.

Capitulo 5 – Transporte e Armazenamento

Um CoDec de vídeo é raramente utilizado sozinho. Constitui quase sempre uma parte de um sistema complexo de comunicação que envolve codificação de vídeo, áudio e respectiva informação. Combinando toda a informação de vídeo e de áudio codificada o sistema de comunicação armazena-a ou transmite-a num *stream*. Existem várias opções para combinar (*multiplexing*), transportar e armazenar dados multimédia codificados. Nos últimos anos tem-se tornado evidente que não existe uma única solução para cada cenário.

5.1 – Mecanismos de Transporte

Nem a norma MPEG-4 nem em particular o H.264 definem um mecanismo de transporte para sinais de vídeo codificados. Contudo, existem várias soluções possíveis dependendo sempre do método de transporte.

1. ***MPEG-2 SYSTEMS***: a 1ª parte da norma MPEG-2 define dois métodos de multiplexagem áudio, vídeo e informações conexas em *streams* apropriados para transmissão (*Program Streams* ou *Transport Streams*). Cada fonte de dados ou *elementary stream* (vídeo codificado ou sequência de áudio), é empacotada em *Packetised Elementary Stream* (PES). Pacotes PES provenientes de diferentes *streams* elementares são multiplexados juntos, de forma a formarem um *Program Stream* (geralmente com um único conjunto áudio e vídeo, como se de um canal de TV convencional se trata-se) ou um *Transport Stream* (que pode conter múltiplos canais) (Figura 49).

   O *Transport Stream* adiciona *Reed-Solomon* e codificação convencional de controlo de erros, para produzir de certa forma protecção a erros de transmissão. O controlo de tempo e sincronismo é apoiado por um sistema de referência de relógio.

   Um *stream* MPEG-4 Visual pode ser processado como um *stream* elementar dentro de um MPEG-2 *Program/Transport Stream*. O transporte de um *stream* MPEG-4 parte 10/H.264 é coberto pela terceira emenda ao MPEG-2 *Systems* [5][29].

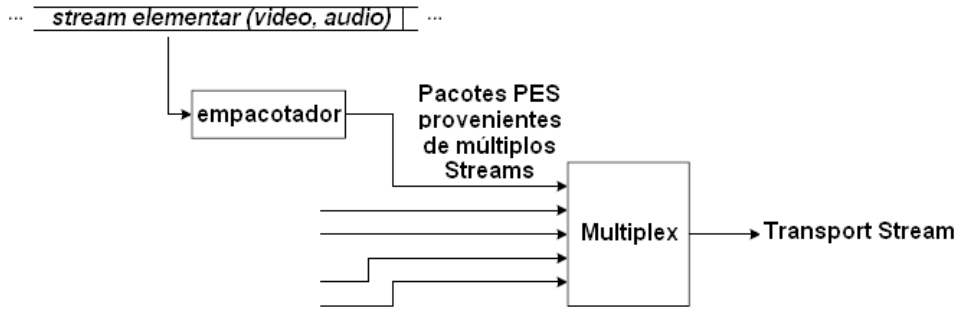

Figura 49 - MPEG-2 *Transport Stream* (adaptada de [5])

Nota: ***Reed-Solomon*** (algoritmo corrector de erros): é um código de correcção de erros que funciona por *oversampling* a um polinómio construído a partir dos dados. O polinómio é avaliado em vários pontos, e esses valores são enviados ou gravados. Amostrando o polinómio mais do que necessita provoca que o mesmo se encontre sob-amostrado. À medida que se recebe muitos pontos correctamente, o receptor pode recuperar o polinómio original apesar da presença de algumas amostras erradas.

2. ***Real-Time Protocol*** (RTP): é um protocolo de encapsulamento (empacotamento) que pode ser usado em conjunto com o *User Datagram Protocol* (UDP) para o transporte de multimédia em tempo real através de redes que utilizem o protocolo de rede IP. O UDP é preferível ao *Transmission Control Protocol* (TCP) para aplicações em tempo real, uma vez que oferece uma latência baixa no transporte em redes IP. No entanto não possui mecanismos de recuperação de sincronismo ou de perda de pacotes. O RTP define uma estrutura de um pacote de dados em tempo real (Figura 50), que inclui um identificador tipo (usado para identificar o tipo de CoDec usado para gerar os dados), um número sequêncial (essencial para reordenamento dos pacotes que são recebidos fora da ordem) e um identificador temporal (*Timestamp*) (necessário para determinar o tempo exacto para a representação dos dados descodificados). Transportar um *stream* de uma sequência áudio/vídeo codificada via RTP envolve encapsulamento de cada *stream* elementar nume série de pacotes RTP, intercalando-os e transmitindo-os por toda a rede IP (usando UDP como protocolo base de transmissão). A carga de dados do RTP é definida para várias normas de áudio e vídeo, inclusive o MPEG-4 Visual e H.264. A estrutura NAL foi desenhada com encapsulamento eficiente para poder ser colocada num pacote próprio RTP. Mais detalhes em relação ao protocolo de transporte RTP serão abordados no Capítulo 6.

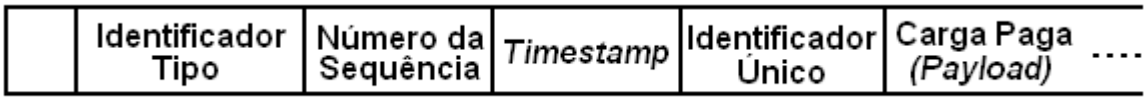

Figura 50 - Estrutura simplificada do pacote RTP (adaptada de [5])

5.2 – Formatos de ficheiro/arquivo

Antigas normas de codificação de vídeo como MPEG-1, MPEG-2 e H.263 não definiam explicitamente um formato para armazenar dados de um arquivo audiovisual. É comum para sequências únicas de vídeo serem armazenadas em ficheiros, necessitando apenas de "mapear" o *stream* codificado numa sequência de bytes. No entanto armazenar e reproduzir sequências combinadas de conteúdos audiovisuais exigem uma estrutura de arquivo mais complexa, especialmente quando, os dados armazenados se encontram a ser difundidos através de uma rede, ou quando o arquivo é necessário para o armazenamento de vários objectos audiovisuais.

O formato de ficheiro MPEG-4 (MPEG-4 *File Format*) e o formato AVC (AVC *File Format*) que se encontram normalizados pela norma MPEG-4, foram concebidos para armazenar conteúdos audiovisuais MPEG-4 e vídeos em formato H.264 respectivamente. Ambos os formatos foram desenvolvidos tendo em conta a *ISO Base Media File Format* que por sua vez é baseada no formato QuickTime da Apple Computer's.

No ISO *Media File Format*, um *stream* codificado (por exemplo uma sequência de vídeo H.264, um objecto de vídeo MPEG-4 Visual ou mesmo um *stream* de áudio) é armazenado como uma faixa, o que representa uma sequência de itens de dados codificados (amostras ou uma *slice* codificada) com referência temporal (Figura 51). Os formatos de arquivo permitem lidar com problemas como é o caso do sincronismo entre pistas, índices de acesso aleatório e transporte do arquivo numa rede de transporte.

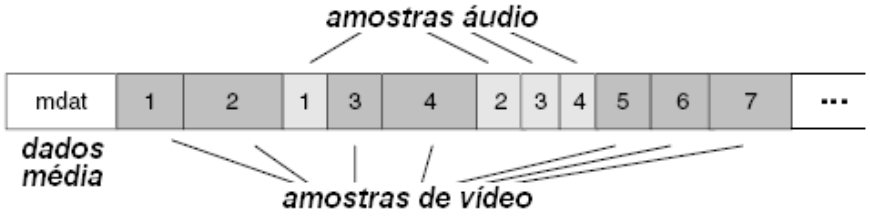

Figura 51 - Ficheiro audiovisual do tipo ISO (adaptada de [5])

O conhecimento do funcionamento do H.264 no que se trata ao transporte e aos formatos de arquivo, leva a que se tomem decisões quanto ao funcionamento de sistemas de conversão e selecção de qualidade de serviço de um cliente. Dito isto, convém saber mais em detalhe o funcionamento da norma MPEG-4 AVC/H.264 no que se trata à transmissão das sequências de vídeo, assim sendo, o capítulo seguinte explicará em detalhe o protocolo de transporte de rede RTP.

Capitulo 6 – Real-Time Protocol (RTP)

O RTP providencia um protocolo de rede *end-to-end* para transmissão de dados em tempo real, tais como áudio e vídeo em serviços de rede *multicast* ou *unicast*. O RTP não reserva e não garante qualidade de serviço (QOS – *Quality Of Service*) para serviços em tempo-real, pelo que o transporte de dados é melhorado através de um protocolo de controlo (RTCP - *Real-Time Transport Control Protocol*). O RTCP permite a monitorização da entrega de dados de forma escalável para grandes redes *multicast*. O RTP e o RTCP são projectados para serem independentes das camadas de transporte da rede.

6.1 – Introdução ao RTP

Com o desenvolvimento da Internet, as actuais políticas de visualização de conteúdos tem vindo a mudar o seu conceito de funcionamento, isto é, tem-se abandonado os conteúdos estáticos em detrimento dos *streams* de vídeo. Habituais conteúdos de texto têm vindo a ser substituídos por música e serviços de voz. Estas alterações requerem novas aplicações que provocam novos desafios para quem desenvolve novas aplicações.

O RTP foi desenvolvido pelo ***Internet Engineering Task Force*** (IETF) no período de 1992-1996 com base no NVP-II (*Network Voice Protocol versão II*), a partir dessa data ferramentas para conferência em domínio *multicast* passaram a utilizar o protocolo RTP como base para a transferência e controlo. A escolha do protocolo RTP não se deveu apenas ao controlo, mas também se deveu a algoritmos de gestão de transmissão de conteúdos, serviços de sincronização e qualidade de serviço que possui.

Dito isto, verifica-se que para transporte de vídeo e áudio em redes IP, o RTP é o padrão a usar. O RTP visa a prestação de serviços e mostra-se útil no transporte de conteúdos audiovisuais sob redes IP, mais precisamente sobre UDP (Figura 52). Estes serviços incluem: recuperação ao nível temporal; detecção de perda de pacotes e sua correcção; identificação de carga (dados) e de fonte; *feedback* de recepção dos dados (útil para ajustes ao nível da transmissão de *stream*); gestão de clientes e sincronismo de conteúdos.

Como se sabe o RTP foi desenvolvido originalmente para o uso em videoconferências *multicast*, mas desde então ele tem-se mostrado muito útil para uma série de outras aplicações, como é o caso da videoconferência em H.323, *webcasting*, distribuição de TV e em telefonia com e sem fios. Para além das aplicações anteriormente referidas o RTP demonstrou um bom desempenho em ligações ponto-a-ponto usadas em sessões *multicast* com várias centenas de utilizadores. Quando foi necessário um protocolo fiável para transmissão de televisão de alta definição (HDTV) de sinais com taxas de

transmissão de alguns gigabits e em situações em que é necessário garantir a qualidade do serviço prestado, o RTP foi o protocolo escolhido [30][31][32].

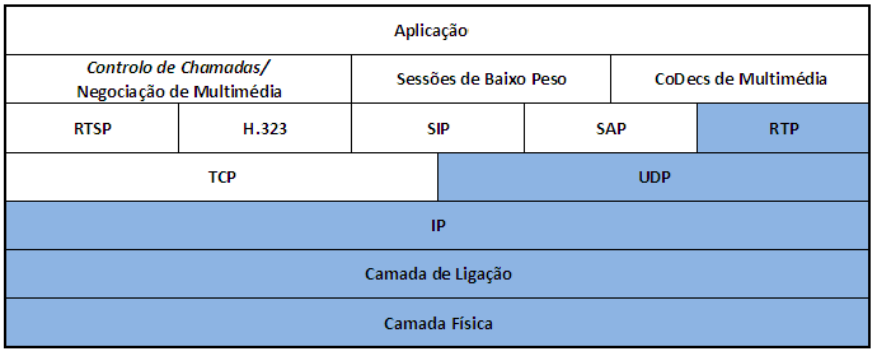

Figura 52 - Pilha Protocolar Multimédia (adaptada de [32])

6.2 – Formato do Pacote RTP

O pacote RTP apresenta as seguintes características (Figura 53)

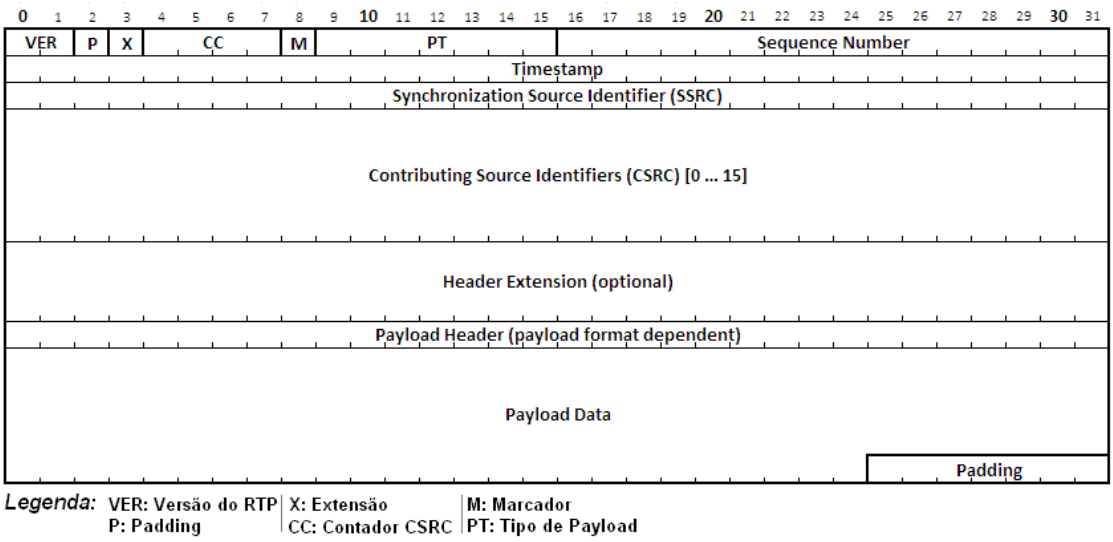

Figura 53 - Estrutura do Pacote RTP (adaptada de [32])

O pacote de dados RTP ilustrado pela Figura 53 pode dividir-se em quatro campos essenciais:

1. Um Cabeçalho RTP (*header*)
2. Uma Extensão ao cabeçalho RTP (*header extension*) (opcional)
3. Um cabeçalho dos dados (*payload header*) (opcional)
4. Os dados a transmitir (*payload data*)

1. Elementos do cabeçalho RTP

O cabeçalho RTP é constituído por um conjunto de 12 octetos, contudo este tamanho pode variar se o pacote RTP contiver uma lista de fontes (*source list*), podendo ser expandido de 4 a 60 octetos adicionais. Os campos principais do cabeçalho RTP são o tipo de *Payload* (*Payload Type*), o número de sequência (*Sequence Number*), *Timestamp* e Identificador de fonte de sincronismo

(*synchronization source identifier*), dentro de outros menos vitais que também o constituem como é o caso dos campos Versão, Marcador, *Padding*, Extensão e Contador CSRC.

Tabela 5 - Descrição do Cabeçalho RTP

Campo	Campo Simplificado	Número de Bits	Função
Versão	VER	2	Identificador da versão do protocolo RTP.
Padding	P	1	Contém informação do número de bytes de enchimento (*padding*) no final do pacote RTP.
Extensão	X	1	Quando colocado a 1, significa que este campo é seguido por um cabeçalho extra (extensão ao cabeçalho).
Contador CSRC	CC	4	O número de CSRC identifica o número de campos CSRC após o cabeçalho fixo do pacote RTP.
Marcador	M	1	Identificador de eventos importantes, como o caso de uma frame fronteira que tem de ser identificada no *stream*.
Tipo de *Payload*	PT	7	Identificador do tipo de dados que se encontra a ser transmitido, serve para a aplicação do receptor identificar o modo de como há-de operar.
Número de Sequência	-	16	Número incremental, identificador de cada pacote RTP enviado para a rede. Pode ser utilizado para identificação de perda de pacotes.
Timestamp	-	32	Reflecte o instante em que se realiza a amostragem do primeiro octeto do pacote RTP. É derivado do relógio que é incrementado de forma monótona e linear no tempo.
Synchronization source identifier	SSRC	32	Identifica a fonte de Sincronismo
Contributing Source Identifiers	CSRC	32	Um *array* de 0 a 15 elementos identificando as fontes da carga paga (*payload*) dentro do pacote RTP. O número de fontes presentes neste campo é dado pelo campo CC.

2. Extensão ao cabeçalho RTP

O protocolo RTP permite realizar extensões ao cabeçalho fixo do protocolo, este campo (opcional) aparece sempre depois do cabeçalho fixo do protocolo e antes do cabeçalho *payload* e do próprio *payload*. Este tipo de campo aparece raramente nos sistemas. É comum o seu aparecimento em sistemas de transmissão em desenvolvimento (em teste), em sistemas que necessitem de um campo adicional no cabeçalho RTP. É de mencionar que este tipo de utilização não é recomendada pela norma do protocolo RTP. A norma recomenda que sempre que uma dada aplicação necessite de um campo adicional o introduza no campo de dados (*payload*) do pacote RTP.

3. Cabeçalho dos dados (*payload header*)

O cabeçalho dos dados é obrigatório no pacote RTP, providencía informação quanto ao *payload*, mas em determinadas aplicações o campo obrigatório não chega para a quantidade de informação que se quer transmitir. Dá-se a necessidade do aparecimento dos cabeçalhos adicionais.

A informação contida no cabeçalho *payload* pode conter informações estáticas – que são as mesmas para toda a sessão ou dinâmicas – que variam de pacote para pacote.

O formato do *payload* irá indicar que partes do cabeçalho são estáticas ou dinâmicas, esta condição tem de ser configurada à cabeça antes do início da sessão.

As partes dinâmicas são normalmente configuradas via *Session Description Protocol* (SDP) [33].

4. Campo de dados (*payload*)

No campo de dados encontra-se a informação que se deseja enviar via protocolo RTP. Neste campo pode-se ter uma ou mais frames, tudo depende da configuração inicial do sistema, de como o programador deseja colocar o protocolo a funcionar.

O formato e o tamanho do campo *payload*, como foi dito, é definido pelo responsável do desenvolvimento da aplicação, dependendo sempre da aplicação e do que se quer fazer com a mesma.

Para sistemas em que se desconhece a forma como o RTP está implementado existem duas formas de identificar o número de frames que se encontram no *payload*:

i. Em casos em que o tamanho da frame é constante, é possível após introspecção do pacote RTP determinar pelo tamanho do pacote o número de frames presentes no campo *payload*.
ii. Outros formatos *payload* incluem um identificador em cada frame encapsulada que indica o tamanho da frame.
    A aplicação do lado do receptor necessita de analisar a frame encapsulada para determinar o número e o início de cada frame. Esta situação é comum quando as frames apresentam tamanho variável.

6.3 – RTCP- Protocolo de Controlo RTP

O protocolo RTP divide-se em duas partes, o protocolo para a transferência de dados, que foi descrito nos pontos anteriores, e o protocolo de controlo de transmissão (RTCP) que providencia relatórios de qualidade de recepção, identificação de participantes na rede, informação necessária para sincronismo de *streams* de dados e outras informações que possam descrever os conteúdos.

A implementação do RTCP divide-se em três partes: o formato do pacote, as regras temporais e a lista dos participantes na rede.

6.3.1 – Formato do Pacote RTCP

Os pacotes RTCP (Figura 54) apresentam todos o mesmo formato que é definido pelos campos:

Version number (V)	Define a versão do protocolo (sempre com o valor dois)
Padding (P)	Indica se o pacote foi preenchido com bits de "enchimento"
Item Count (IC)	Alguns tipos de pacotes contêm uma lista de itens anexa, informação específica de relevo. O campo *Item Count* serve para indicar o número de itens incluídos no pacote
Packet Type (PT)	Identifica o tipo de informação que se encontra a seguir ao cabeçalho do pacote
Lenght (tamanho)	Indica o tamanho que a informação presente no pacote ocupa após o cabeçalho do pacote RTCP. É medido em unidades de 32 bits, pois todos os pacotes RTCP são múltiplos de 32 bits.

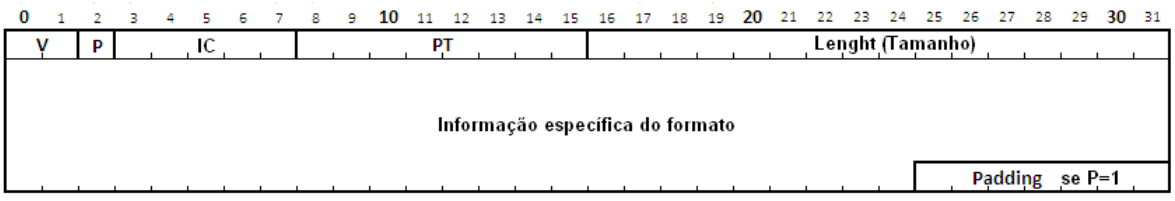

Figura 54 - Formato do Pacote RTCP (adaptada de [32])

6.3.2 – Tipos de Pacote RTCP

Existem cinco tipos de pacotes que se encontram definidos pela norma RTP: *Receiver Report* (RR), *Sender Report* (SR), *Source Description* (SDES), *Membership Management* (BYE) e *Application Defined* (APP).

Em que as suas funções são:

Sender Report

Este pacote contém um relatório de envio e recepção de pacotes RTP, criado pelos participantes na comunicação, ou seja, que participam activamente no envio de pacotes na rede (fontes activas).

Receiver Report

Este pacote contém um relatório do envio e recepção de pacotes RTP dos participantes que não contribuem para o envio de pacotes, isto é, que não são fontes activas da rede.

Source Description Items

Contém um descritivo do participante. Inclui a informação do seu nome canónico (CNAME), nome único na rede.

Membership Management

Indica a saída do participante da comunicação e contém a identificação do SSRC/CSRC do mesmo.

Application Defined

Contém funções específicas da aplicação.

Cada sessão RTP é identificada pelo seu endereço na rede e por um par de portos: um para o RTP de dados e outro para o protocolo de controlo RTCP. O porto do RTP de dados deve ter terminação par e o porto do RTCP deve ser o porto imediatamente a seguir. Por exemplo, se os dados se encontram a ser enviados pelo porto UDP 5004, então o porto para controlo será o porto 5005.

Após o estudo detalhado dos conceitos base do vídeo, sua codificação e transmissão (para o caso particular da norma MPEG-4 AVC/H.264) encontramo-nos em óptimas condições para se proceder ao desenvolvimento do Conversor de vídeo. Deste modo nos próximos capítulos são descritos os desenvolvimentos do projecto bem como a plataforma de desenvolvimento e respectivo software utilizado para tal.

Inicia-se por uma descrição da plataforma de hardware utilizada, passando de seguida para as ferramentas de software. Por último, apresenta-se de uma forma rigorosa o trabalho desenvolvido e respectivos testes ao seu desempenho e operacionalidade.

Capitulo 7 – Plataforma e Ferramentas de Desenvolvimento

Este capítulo tem como objectivo, dar a conhecer a plataforma utilizada no desenvolvimento do presente trabalho, bem como as ferramentas utilizadas para o efeito.

A plataforma de hardware utilizada foi um mini PC com uma versão instalada do sistema de operação *open source*, *Ubuntu*.

O conteúdo do projecto foi desenvolvido utilizando linguagem C, compilando a mesma com o *GNU Compiler Collection* (GCC).

De forma a se ter conhecimento dos conteúdos que circulavam na rede utilizou-se um analisador de protocolos de rede – *Wireshark*. Com esta aplicação passou-se a ter controlo e conhecimento do tipo de conteúdos que circulavam na rede do sistema desenvolvido

7.1 – Plataforma de Hardware – Mini PC

A plataforma utilizada como base deste projecto foi o mini PC apresentado na Figura 55.

Figura 55 - Mini PC

Esta plataforma foi escolhida por apresentar dimensões reduzidas (23 x 18 x 5 cm) e por possuir três placas de rede *Ethernet* (Lan1, Lan2, Lan3), como se pode ver na Figura 56. Torna-se importante a presença deste tipo de característica dado que o sistema a ser

desenvolvido, irá funcionar como um "filtro" aos pacotes RTP existentes na rede, pacotes estes que possuem os conteúdos de vídeo que o cliente pretende visualizar.

Para além das três placas de rede este computador possui ainda três entradas USB (duas na face posterior e uma na face anterior do mini PC) (Figura 56 e 57), que podem ser utilizas para expansão da plataforma, podendo-se aumentar ainda mais o número de placas de rede no computador, ideal para casas de clientes que possuam mais de um aparelho de visualização de conteúdos de vídeo, bem como para ligação de outro tipo de periféricos (drives de CD's e de cartões memória).

Figura 56- Face Posterior da plataforma

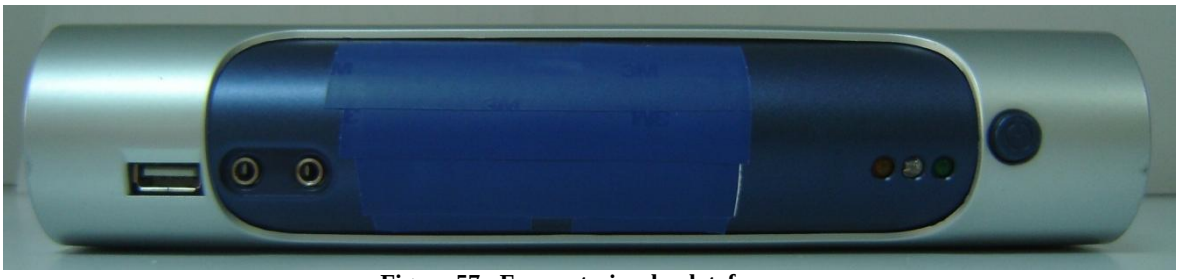

Figura 57 - Face anterior da plataforma

Para além das características que se podem visualizar à partida, pela visualização das imagens anteriores, sabe-se ainda que esta plataforma possui um processador *Via Erza* a 800 MHz, uma memória de 256 MBytes e um disco de 60 GBytes. Sabe-se que possui uma placa gráfica (32bits com 8MB dedicados) e uma placa de som (de 32 bits), ambas integradas na *motherboard* da plataforma.

7.2 – GCC – *GNU Compiler Collection*

O GCC [34], é um projecto de computação, iniciado por *Richard Stallman* em 1984, com o intuito de criar um sistema de operação totalmente livre, que qualquer pessoa possa usar, modificar e redistribuir sem ter de se preocupar com licenças de utilização e distribuição. A única condição imposta pelo criador é a existência de compatibilidade com o sistema de operação *UNIX*.

A partir de 1984, *Stallman* e outros programadores que abraçaram a causa, foram desenvolvendo os módulos principais do sistema de operação, como é o caso do compilador C e dos respectivos editores de texto.

Em 1991, o sistema de operação encontrava-se já na sua fase final, mas continuava a faltar uma parte muito importante, o *kernel* (o cerne do sistema de operação, a camada de software mais próxima do hardware). *Stallman* e a sua equipa encontravam-se a desenvolver um *kernel*, de seu nome *Hurd*, quando aconteceu algo que mudou o rumo da história, um jovem de seu nome *Linus Torvalds*, havia desenvolvido um *kernel* que podia usar todas as peças do sistema *GNU*, este *kernel* ficou conhecido como *Linux*, contracção do nome *Linus* com *UNIX* [34].

Actualmente o sistema *GNU* vem integrado na maioria das distribuições de *Linux* como é o caso do *Ubuntu* utilizado no desenvolvimento deste trabalho.

7.3 – *Wireshark*

O *Wireshark* [35] (muito conhecido como *Ethereal*, nome mudado em 2006 devido a problemas com os direitos de autor do nome anterior) é um analisador de protocolos de rede, de uso gratuito.

Este analisador foi desenvolvido por um grupo de peritos ao nível global e é produto de um projecto de escala global iniciado em 1998.

De forma simples o *Wireshark*, permite realizar uma análise do comportamento da rede, bem como a resolução de problemas de utilização de protocolos de comunicação em desenvolvimento. Nos últimos anos tem-se vindo a tornar um dos analisadores de rede mais utilizados, de facto um dos mais importantes ao nível da indústria e do ensino.

No desenvolvimento deste trabalho esta aplicação foi muito importante ao nível do aperfeiçoamento dos *links* de rede entre as diversas plataformas de transmissão de conteúdos de vídeo, dado que permite a análise dos conteúdos transmitidos, utilizando o protocolo RTP (Figura 58).

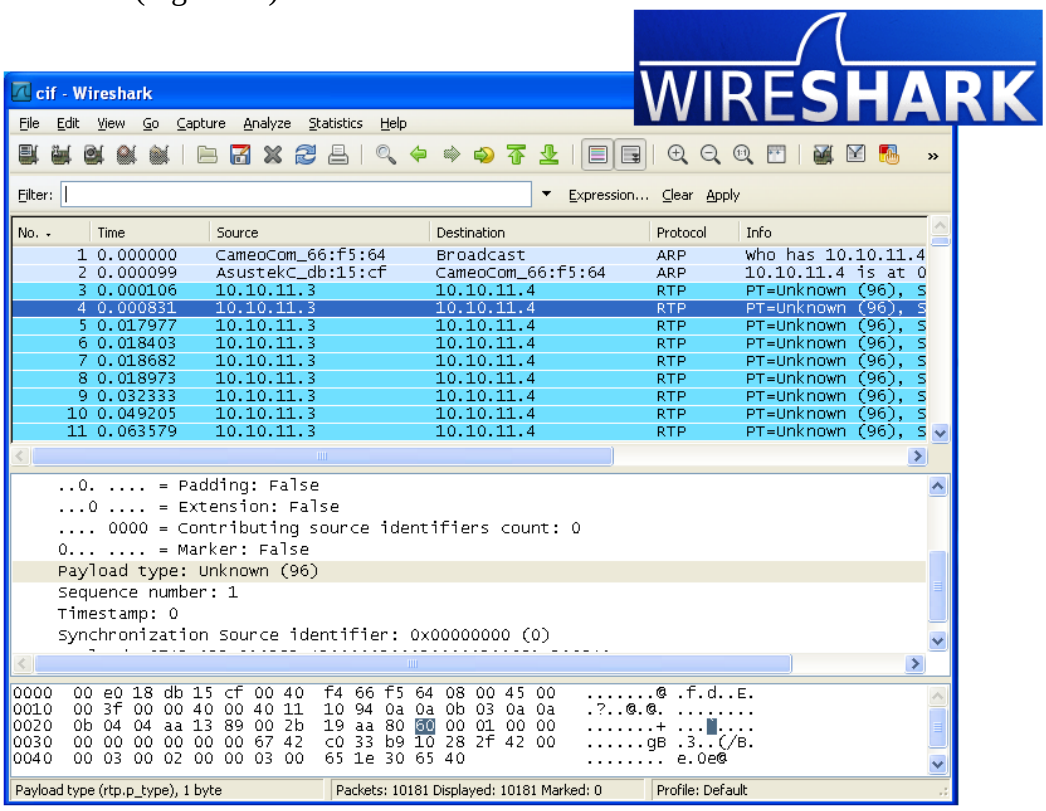

Figura 58 - Aplicação *Wireshark*

7.4 – *HexEdit*

O *HexEdit* [36] é uma aplicação *freeware* de leitura e edição de ficheiros em formato hexadecimal para as plataformas *Windows* e *Linux*.

Esta aplicação permite ao utilizador ver e editar qualquer tipo de ficheiro não interessando o tipo de formato no qual se encontra gravado o ficheiro. O *Hexedit* permite ainda cortar, copiar, colar, inserir e eliminar qualquer quantidade de informação, desde que nunca se ultrapasse o limite máximo de tamanho de ficheiro de 4 Gb.

Esta aplicação torna-se muito importante para o desenvolvimento deste trabalho dado que para a verificação de envio e recepção de pacotes se torna necessária para verificar se o conteúdo do pacote enviado do transmissor para o cliente é alterado de alguma forma com a interacção na rede (Figura 59).

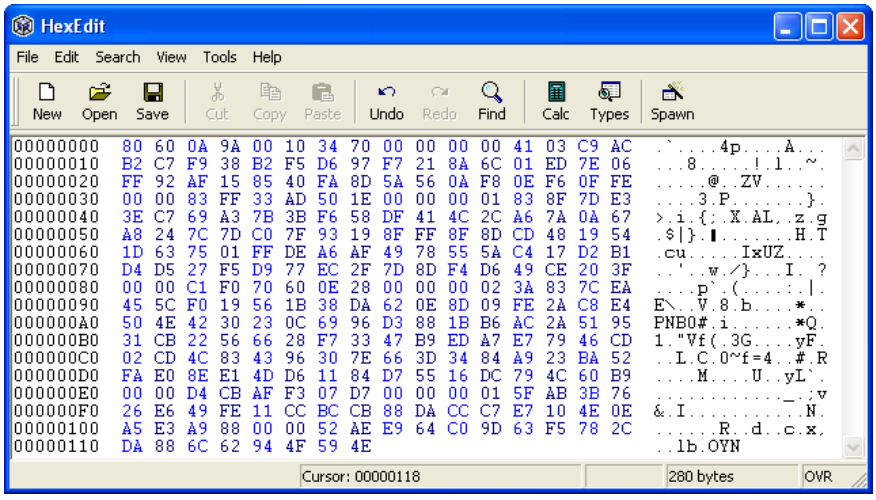

Figura 59 - Aplicação *HexEdit*

Capitulo 8 – Conversor de Formatos de Vídeo

Este capítulo tem como objectivo dar a conhecer a plataforma desenvolvida no âmbito da realização do presente trabalho científico. É apresentado o algoritmo e a respectiva implementação do Sistema de Conversão de Vídeo desenvolvido. São apresentados ainda dois demonstradores criados com o intuito de testar o bom funcionamento do Conversor de Vídeo, bem como os resultados obtidos na transmissão dos diversos formatos de vídeo.

Nota: Este trabalho foi integrado com outro trabalho científico, com o intuito de realizar uma demonstração do seu funcionamento em caso real. A parte do sistema de comunicações é da autoria de Luís André Vela dos Reis integrada no seu trabalho de mestrado de nome "Empacotamento, multiplexagem e Encriptação para TV" realizada em 2008.

8.1 – Conceito/Algoritmia

O conceito base para o desenvolvimento deste projecto foi pensar numa máquina de estados de comportamento simples, isto é, a máquina (Conversor de Formatos) recebe um pacote RTP, testa o seu "conteúdo" (como se sabe não é o conteúdo do pacote RTP que é testado mas sim o SSRC do mesmo) e caso o SSRC seja coincidente com o serviço a que o cliente tem acesso (CIF, SD ou HD), esse pacote é enviado para a saída do conversor. Desta forma o cliente recebe unicamente os pacotes a que tem direito (Figura 60).

Figura 60 - Conceito base do conversor de formatos (Sistema de Conversão de Vídeo)

<u>Algoritmo Implementado</u>:

i. Recepção do Pacote RTP;

ii. Teste aos dois primeiros Bytes para verificação de início e validade de pacote RTP;

iii. Se os dois primeiros Bytes forem diferentes de 0x80 (1º Byte) e 0x60 (2º Byte), o pacote recebido é descartado e espera pela recepção de novo pacote. Segundo Byte corresponde ao PT do pacote RTP, se igual a 96, identifica o sistema de transmissão de pacotes RTP – *SUIT* [37];

iv. Se os dois primeiros Bytes forem iguais a 0x80 (1º Byte) e 0x60 (2º Byte), é realizada uma leitura aos 12 Bytes iniciais do pacote RTP;

v. Testa-se o 9º, 10º, 11º e 12º Bytes lidos no pacote RTP (teste ao valor existente no SSRC do pacote RTP);

vi. Caso o SSRC assuma o valor 0x0000, o pacote recebido é identificado como sendo um pacote pertencente ao serviço CIF. Se o SSRC assumir o valor 0x0001 então o pacote pertence ao serviço SD. Caso o SSRC do pacote RTP tome o valor 0x0002 então encontramo-nos numa situação de transmissão de um serviço HD;

vii. O valor do SSRC lido no pacote recebido é testado com o valor introduzido pelo administrador do sistema quando se dá a contratação do serviço por parte do cliente (por exemplo, se o cliente contratar serviço de recepção de vídeo CIF, só recebe o formato CIF, logo não pode receber os outros formatos em sua casa);

viii. Caso se verifique que o pacote recebido possui um valor de SSRC coincidente com o serviço contratado, então o pacote recebido é copiado para a saída do conversor e é transferido para o cliente;

ix. O Conversor passa a um estado de espera até nova recepção de pacotes RTP. No momento em que o Sistema de Conversão de Vídeo recebe novo pacote RTP, inicia-se novo ciclo do algoritmo.

8.2 – Implementação e Código Desenvolvido

Baseando-nos no algoritmo anterior partiu-se para a implementação prática do sistema, recorrendo às ferramentas GNU de desenvolvimento.

Com os conhecimentos de linguagem C, criaram-se dois demonstradores. Um para um único serviço, isto é, para o envio, recepção e tratamento de um pacote RTP único. O segundo demonstrador serve para o estudo da resposta ao envio de vários pacotes RTP com vários serviços associados.

Dada a complexidade do código criado e para a melhor compreensão do Sistema de Conversão de Vídeo desenvolvido, apresenta-se abaixo um fluxograma representativo do mesmo (Figura 61). Encontra-se anexado a este trabalho científico, um CD intitulado "Sistema de Conversão de Vídeo – Conteúdos Práticos" com os conteúdos desenvolvidos (Página 133 e 134).

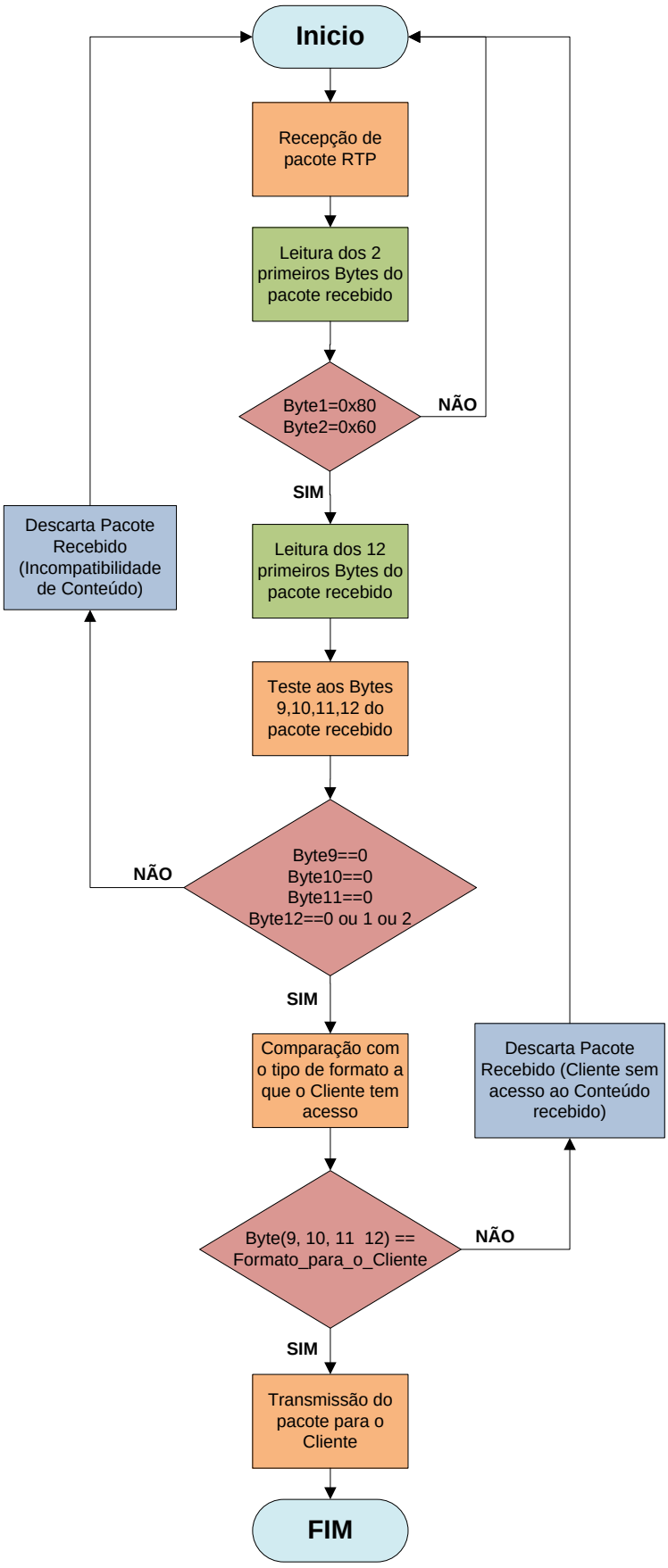

Figura 61 - Fluxograma do Conversor de Vídeo desenvolvido

8.3 – Testes de Comportamento e Desempenho

Demonstrador 1: Envio de um pacote RTP com o consentimento do operador do sistema

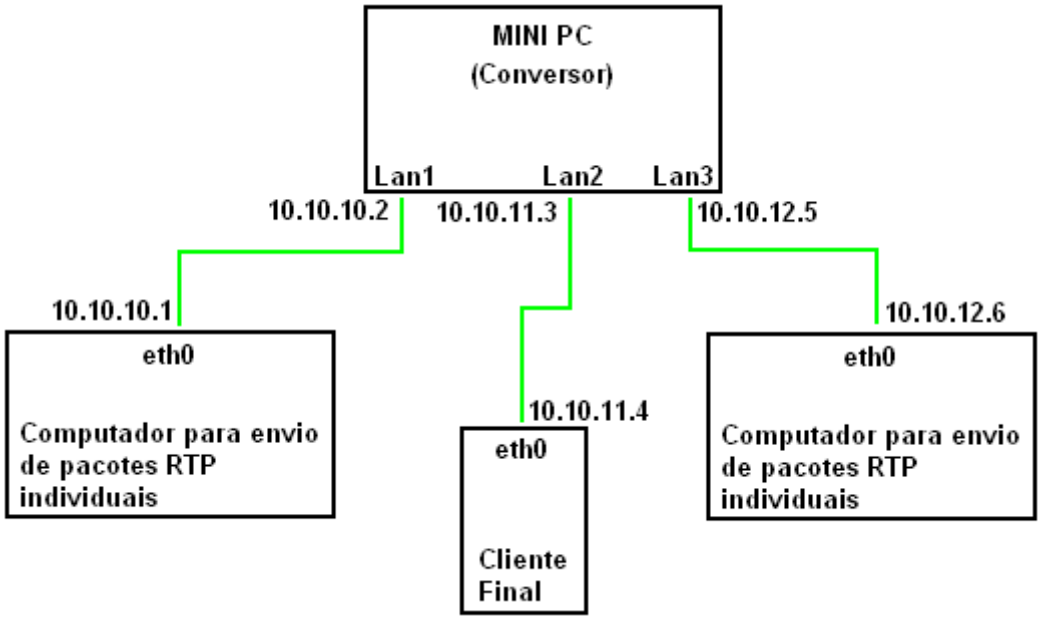

Figura 62 – Demonstrador 1 de funcionamento do Conversor de formatos

Este demonstrador (Figura 62) foi desenvolvido para explicar o comportamento assumido pelo conversor no tratamento de um pacote RTP. Este demonstrador realiza o estudo do pacote RTP quanto ao seu tipo, isto é, testa o início do pacote RTP (1º Byte igual a 0x80) e o PT do pacote para ver se este coincide com o do nosso sistema (2º Byte do pacote RTP, com o valor 0x60). PT igual a 0x60 corresponde ao identificador 96 em decimal. Este valor identifica o nosso sistema de envio de pacotes.

Realiza ainda um teste quanto ao serviço que se encontra a ser transmitido, isto é, compara o valor do SSRC do pacote RTP recebido com o valor correspondente ao serviço autorizado ao cliente do sistema. Sempre que recebe um pacote RTP do nosso sistema de transmissão imprime uma mensagem de identificação de pacote pertencente ao nosso sistema (identificação essencial para redes em que se encontram instalados sistemas de transmissão semelhantes ao nosso). Sempre que o Conversor recebe um pacote coincidente com o serviço autorizado ao cliente, este demonstrador imprime uma mensagem identificativa do tipo de serviço a que o cliente tem acesso. Para além desta mensagem imprime outra a garantir que existiu a devida transmissão do pacote recebido para o cliente.

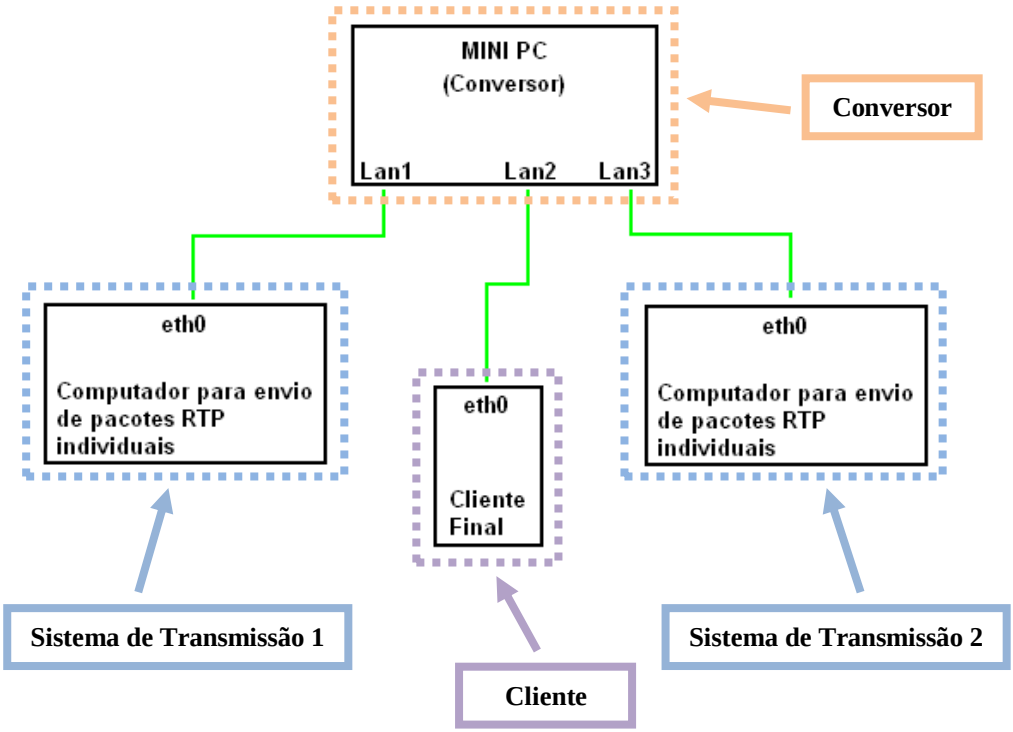

Figura 63 - Divisão em Secções da Rede Desenvolvida

<u>Para se lançar este demonstrador, deve fazer como o indicado:</u>

i. <u>No cliente</u>:

./Cliente <Porta do Conversor>

ii. <u>No Conversor</u>:

./Conversor <Porta do Servidor> <Tipo de Ficheiro> <Endereço do Cliente> <Porta do Cliente>

iii. <u>No Sistema de Transmissão de Pacotes RTP</u>:

./Servidor <Endereço do Conversor> <Serviço a Enviar> <Porta do Conversor>

<u>NOTA:</u> Para visualizar as imagens do lançamento de cada aplicação do demonstrador, ver Anexo 7.

Modo de Funcionamento:

O Computador para envio de pacotes RTP, envia um pacote mediante a autorização do operador (Figura 65) para o conversor, fluxo a vermelho da Figura 64.

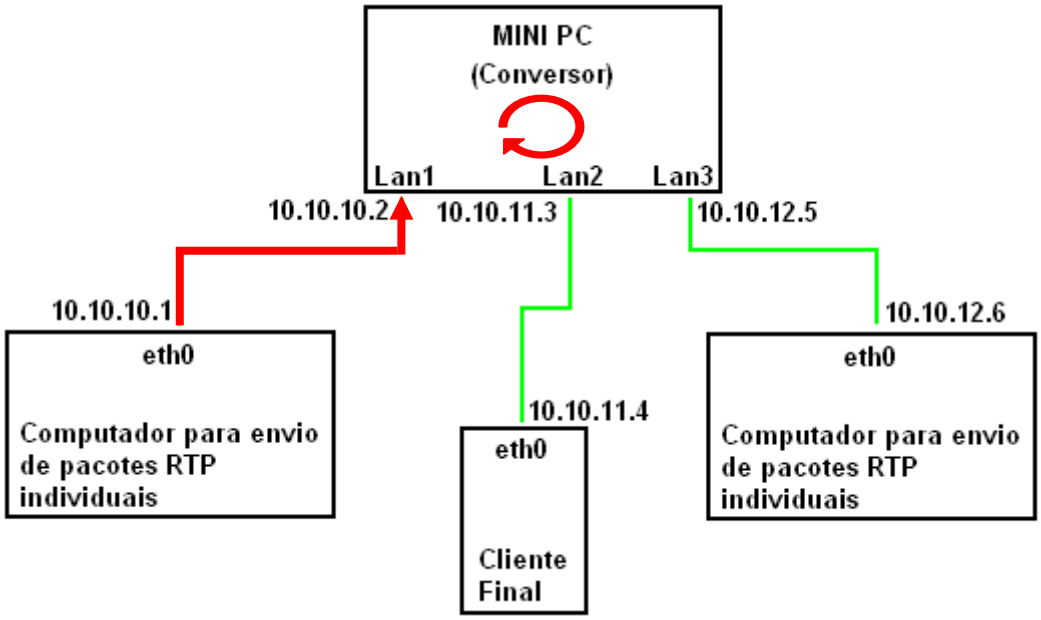

Figura 64 - Envio de pacote RTP do Transmissor para o Conversor

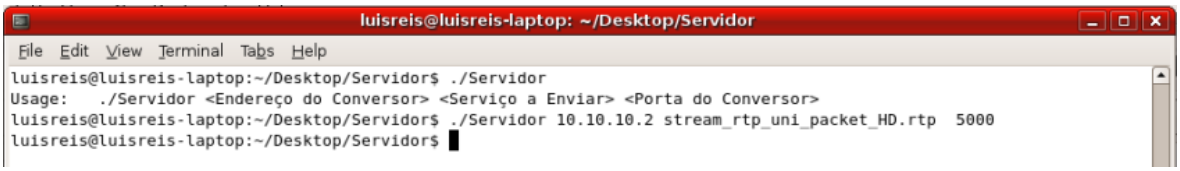

Figura 65- Envio de pacote RTP, no servidor, com serviço HD

O Conversor lida com a recepção do pacote, emitindo as mensagens de tratamento do mesmo (mensagens de identificação de pacote pertencente ao nosso sistema e de identificação do serviço autorizado caso nos encontremos nessa situação) (Figura 66).

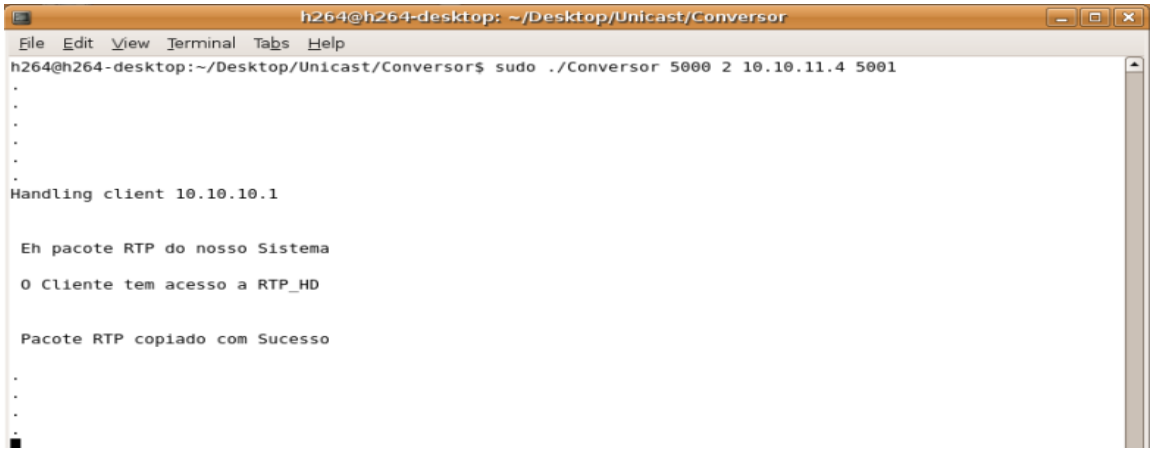

Figura 66 - Tratamento de pacote HD no conversor

Quando o pacote enviado coincide com o formato autorizado ao cliente, o conversor envia o pacote para o cliente, como o indicado pelo fluxo vermelho na Figura 67.

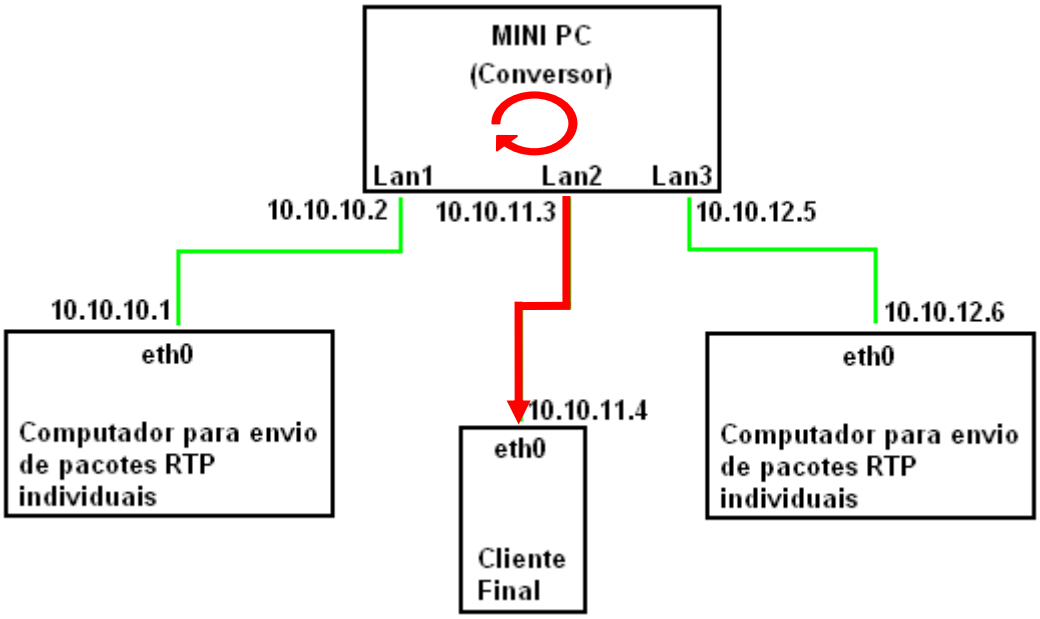

Figura 67 - Envio do pacote RTP recebido do Conversor para o Cliente

Como se pode verificar, pelas Figuras 68 e 69, não existiu perda do pacote na rede (visto que alcançou o destino – o cliente) nem alteração do seu conteúdo. As conclusões que se retiraram serviram para atestar o bom funcionamento do sistema desenvolvido para a situação em que circula um único pacote RTP do nosso sistema na rede IP criada neste demonstrador.

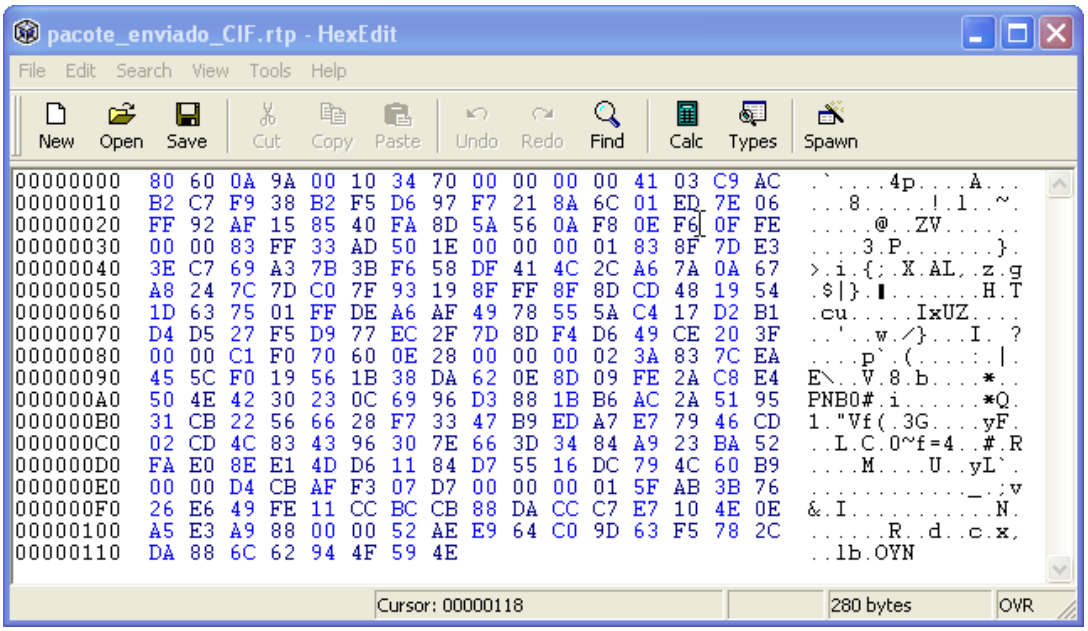

Figura 68 - Pacote enviado do Transmissor 1 para o Conversor (Visualizado no Sistema de Transmissão 1)

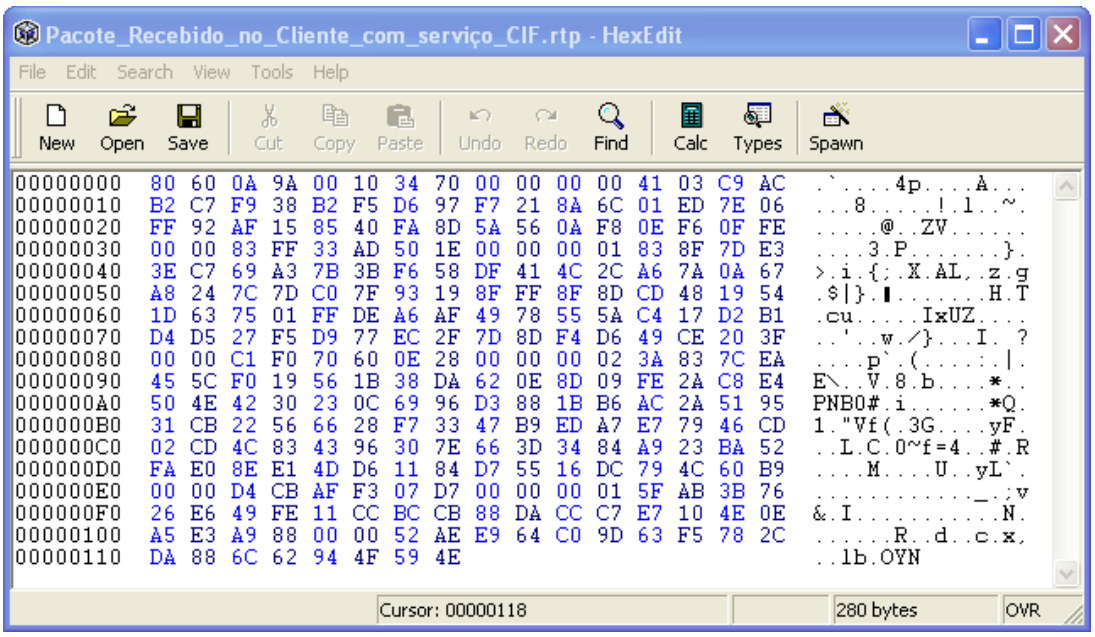

Figura 69 - Pacote Enviado do Conversor para o Cliente (Visualizado no Cliente)

Demonstrador 2: Envio de *stream* multi-serviço de pacotes RTP

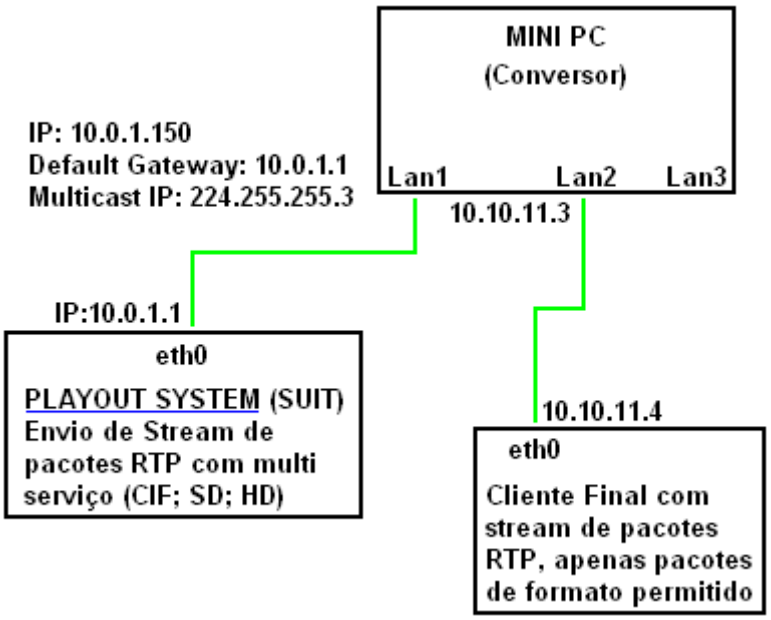

Figura 70 - Demonstrador 2 de funcionamento do Conversor de formatos

Este demonstrador (Figura 70) foi criado para exemplificar o modo de funcionamento do conversor quando estamos perante a transmissão de vários serviços em simultâneo na nossa rede. Este caso exemplifica a situação em que se encontram vários clientes ligados na mesma rede, tendo cada um acesso a um serviço diferente. Este trabalho permite o estudo mais detalhado do funcionamento da nossa rede com vários serviços a

decorrer na mesma. Assim permite-nos estudar o comportamento do conversor desenvolvido quando se encontra introduzido numa rede multi-serviço.

Para lançar este demonstrador deve fazer como indicado:

i. No cliente:

./Cliente <Porta do Conversor>

ii. No Conversor:

./Conversor_Multicast <Endereço Multicast> <Porta Multicast> <Tipo de Ficheiro> <Endereço do Cliente> <Porta do Cliente>

iii. PLAYOUT SYSTEM (SUIT):

mcsend (Esta aplicação não se encontra incluída no CD, pois foi cedida apenas para testes em laboratório)

NOTA: Imagens do lançamento de cada aplicação do demonstrador no Anexo 8.

Modo de Funcionamento:

O PLAYOUT SYSTEM (*SUIT*) [37] envia vários pacotes RTP para o conversor, pertencendo estes a vários serviços de TV (CIF, SD e HD). O conversor recebe um pacote RTP e lida com este como no demonstrador 1, isto é, recebe o pacote RTP, se coincidente com o serviço a que o cliente está autorizado a receber então envia o mesmo para o cliente, senão, descarta o pacote recebido e prepara a recepção de outro (Figura 71 e 72).

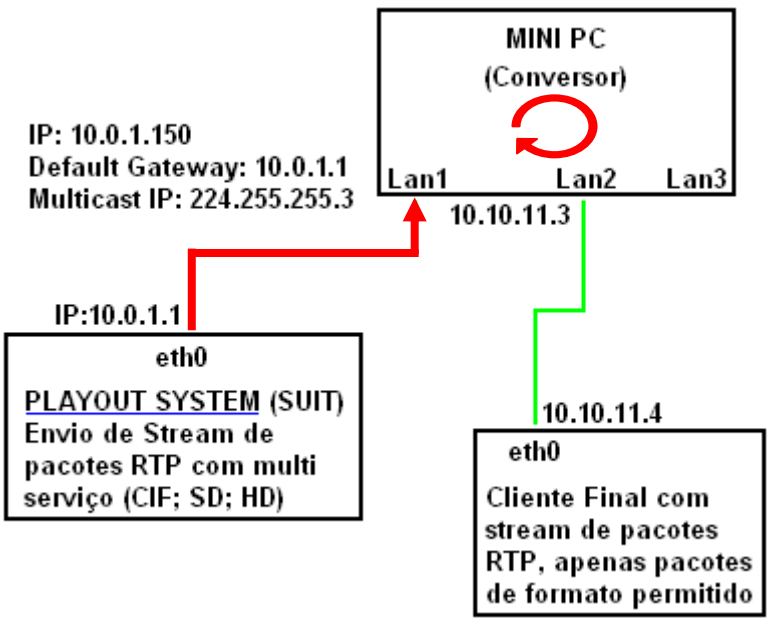

Figura 71 - Envio de pacotes RTP do PLAYOUT para o Conversor

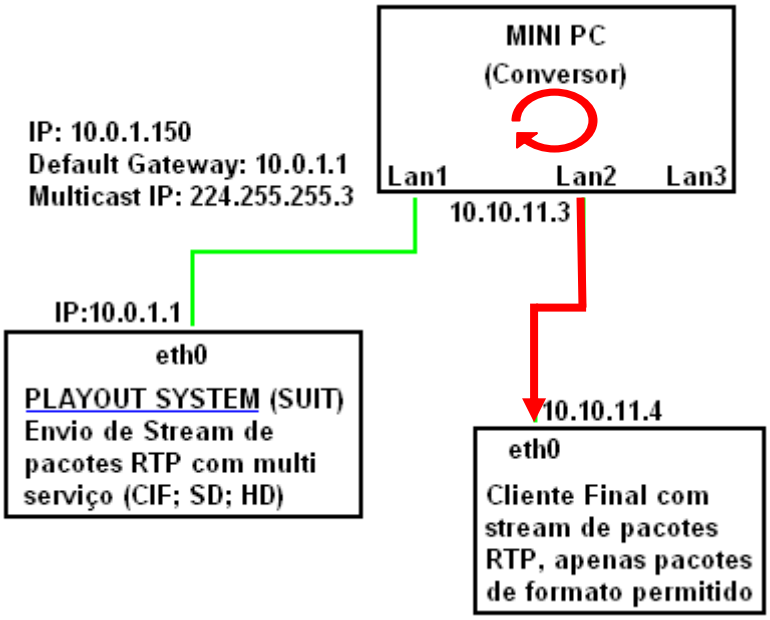

Figura 72 - Envio dos pacotes RTP recebidos do Conversor para o Cliente

Com o estudo deste demonstrador, pode-se analisar o comportamento do Conversor quando nos encontramos numa rede em que circulam todos os serviços em simultâneo. Foram realizados estudos para o acesso do cliente a serviços CIF, SD e HD.

Para cada serviço CIF, SD e HD, e utilizando a ferramenta de análise de protocolos de rede *Wireshark*, foi realizada a captura dos pacotes que chegavam ao cliente. Sendo posteriormente comparado o número de pacotes que alcançaram o destino com o enviado pelo servidor.

Como se pode verificar na Figura 73, são enviados pelo servidor de cada vez que se rearma o PLAYOUT SYSTEM, 10177 pacotes RTP correspondendo ao serviço CIF, 11691 correspondentes ao serviço SD e 22623 pacotes correspondentes ao serviço HD.

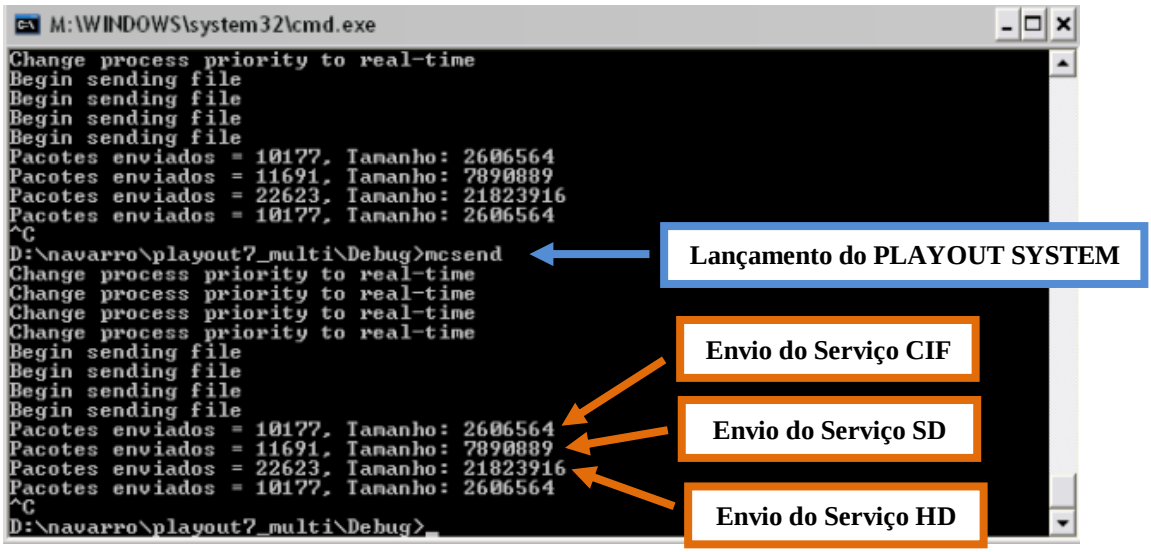

Figura 73 - Envio de pacotes RTP com multi-serviço no PLAYOUT

Serviço CIF:

Com o *Wireshark* foi realizada a captura (Figura 74) dos pacotes enviados pelo PLAYOUT. É de lembrar que esta captura foi realizada no cliente, ou seja, após a passagem/selecção dos pacotes RTP no Conversor.

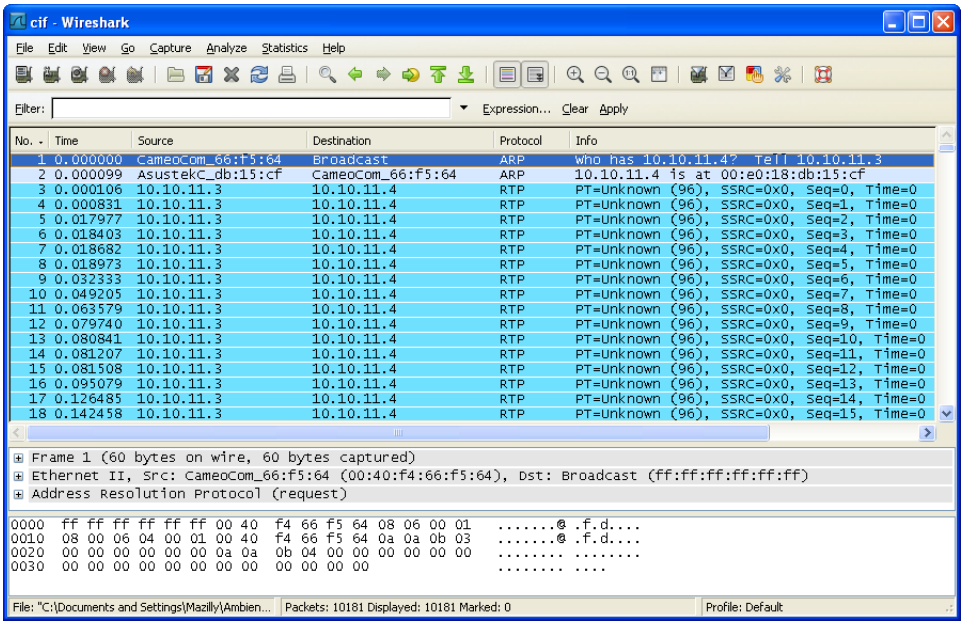

Figura 74 - Captura dos pacotes RTP enviados pelo PLAYOUT, para o serviço CIF

Da captura dos pacotes realizada, analisou-se a recepção do *stream* enviado com o mesmo programa, no qual se obteve a resposta (Figura 75)

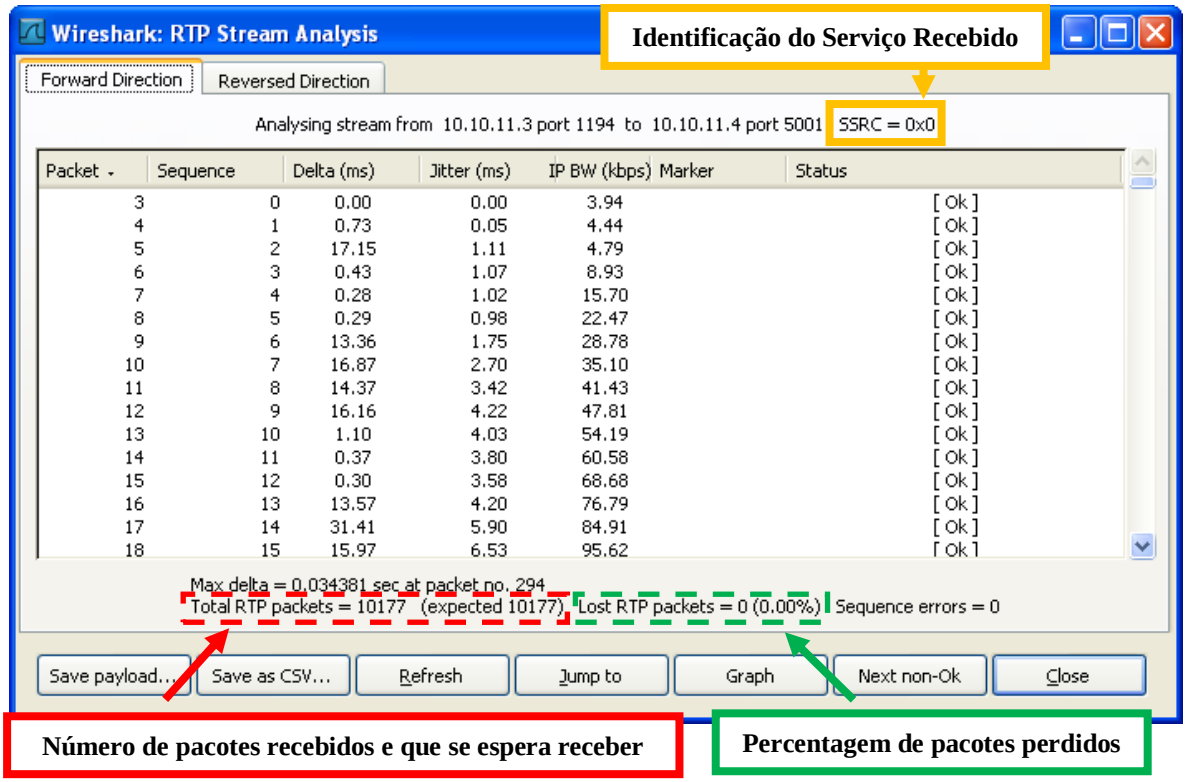

Figura 75 - Análise estatística da recepção do serviço CIF no cliente

Serviço SD:

Com o *Wireshark* foi realizada a captura (Figura 76) dos pacotes enviados pelo PLAYOUT para o cliente.

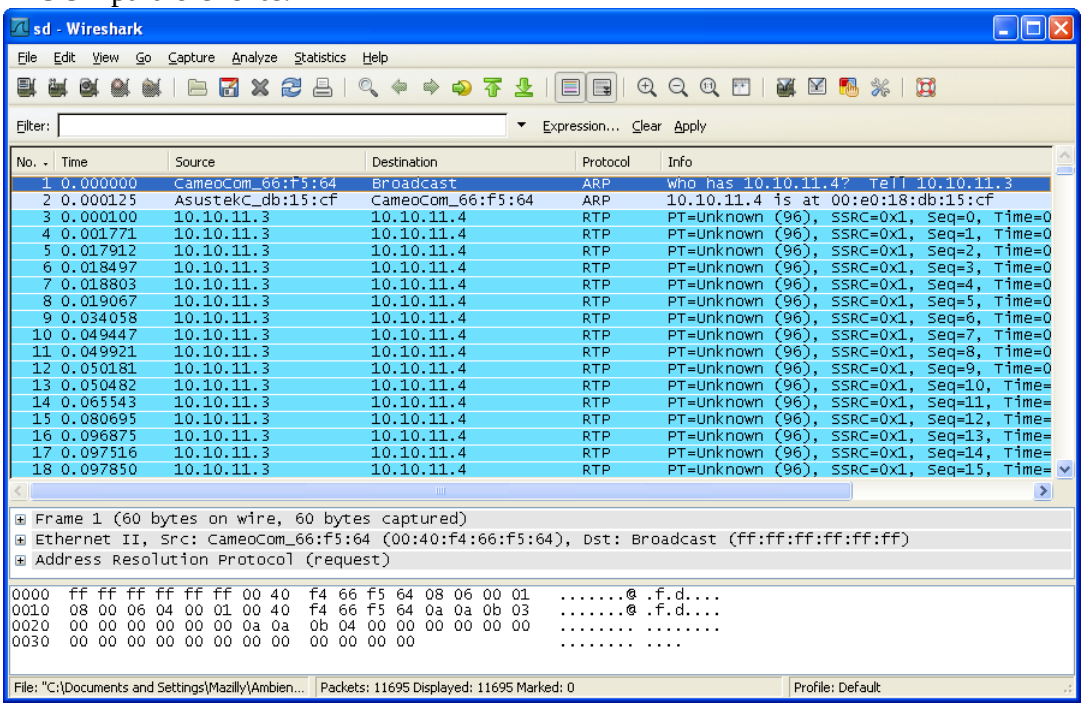

Figura 76 - Captura dos pacotes RTP enviados pelo PLAYOUT, para o serviço SD

Da captura anterior, resulta a seguinte análise ao *stream* (Figura 77).

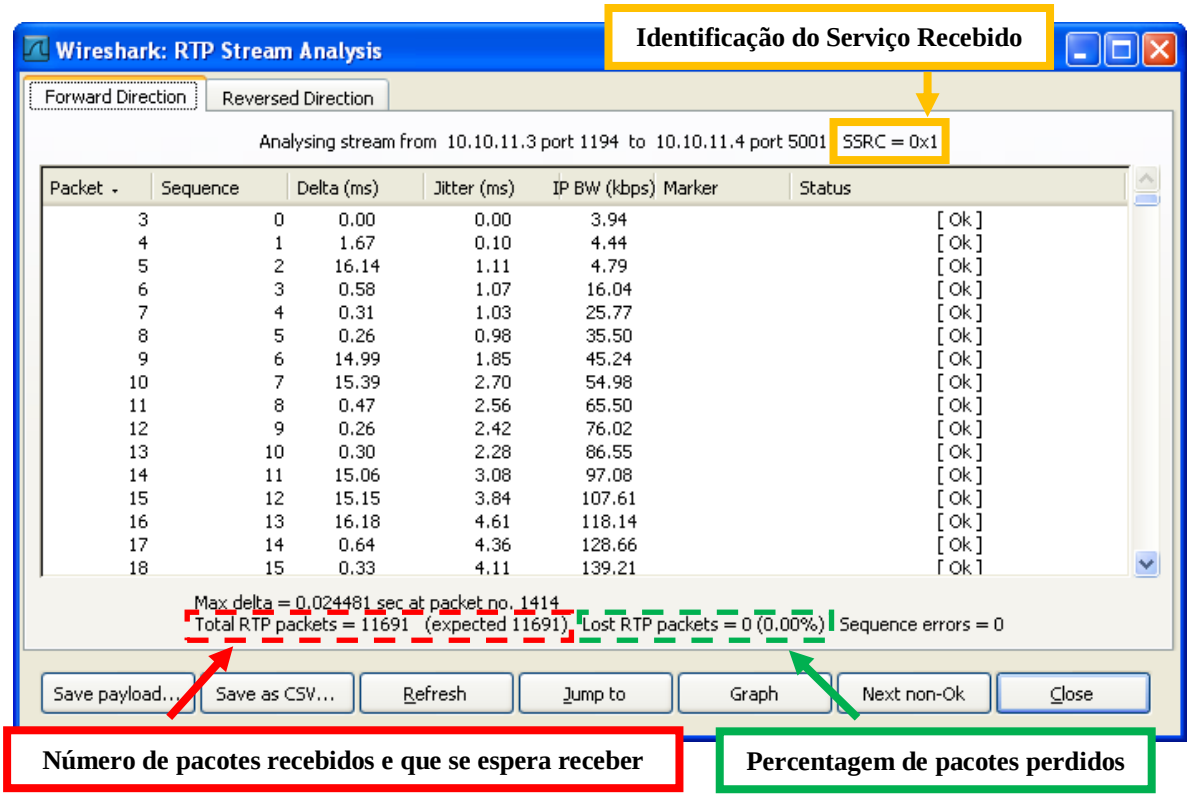

Figura 77 - Análise estatística da recepção do serviço SD no cliente

Serviço HD

Com o analisador de protocolos realizou-se a captura (Figura 78) dos pacotes enviados pelo PLAYOUT para o cliente.

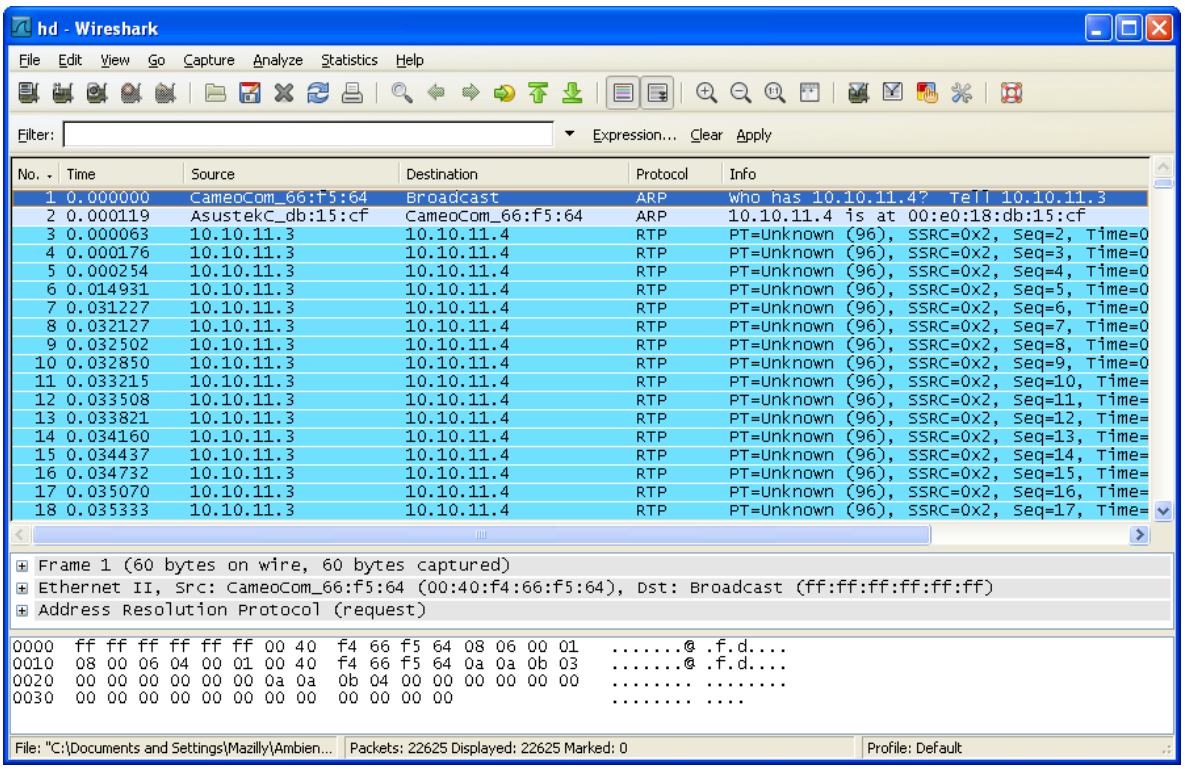

Figura 78 - Captura dos pacotes RTP enviados pelo PLAYOUT, para o serviço HD

Da captura anterior, resulta a análise ao *stream* (Figura 79).

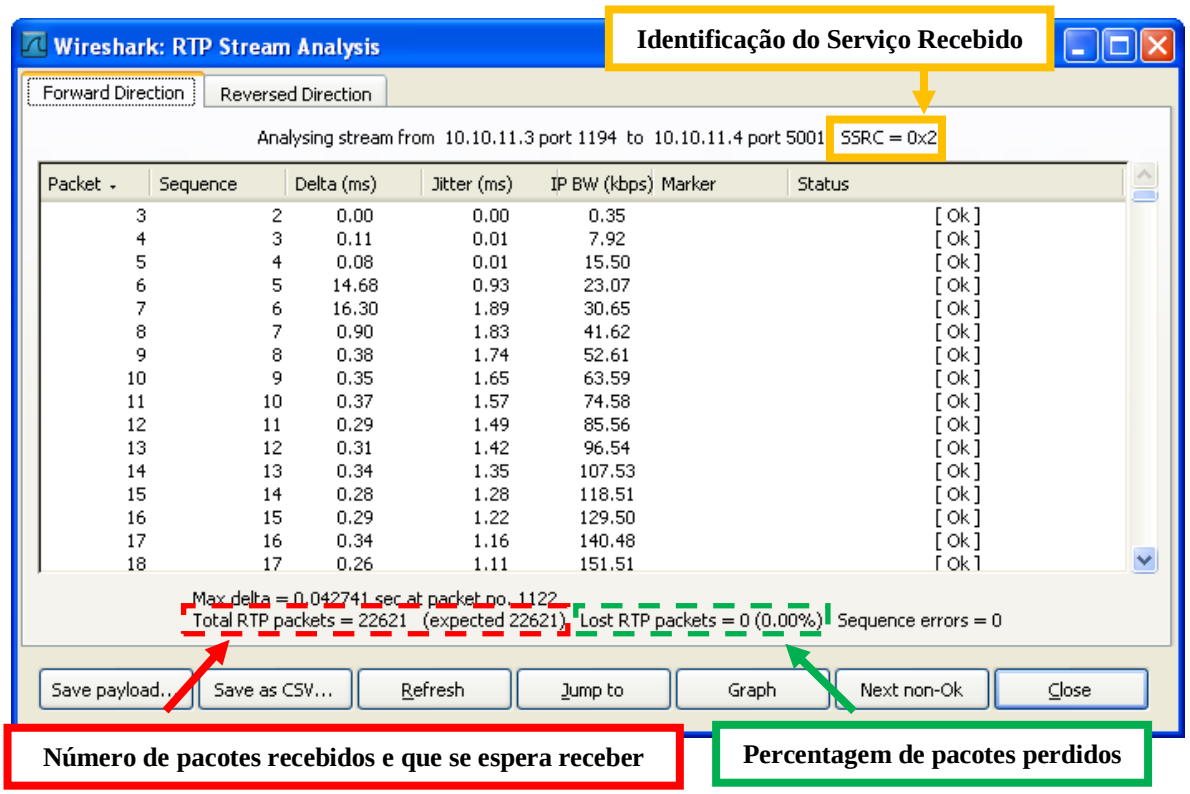

Figura 79 - Análise estatística da recepção do serviço HD no cliente

Como se pode verificar pela análise das figuras anteriores, o Conversor apresenta um comportamento linear, isto é, o seu comportamento não é afectado com o aumento do número de serviços que se encontram a ser enviados para a rede. Podemos fincar a nossa posição com os resultados obtidos para os três serviços (CIF, SD e HD) existentes na rede do demonstrador 2.

Todos os pacotes enviados pelo PLAYOUT alcançam o destino desejado, ou seja, alcançam o cliente final quando este tem o respectivo serviço contratado.

Este resultado era o esperado, uma vez que o tratamento dos pacotes RTP se dá sempre da mesma forma. O serviço a que o cliente tem acesso é testado logo á cabeça antes de ser realizado qualquer outro tipo de tratamento ao pacote, evitando-se assim perdas de tempo desnecessárias que poderiam provocar perdas de pacotes na rede.

Capitulo 9 – Conclusões

A codificação de vídeo tem inúmeras aplicações, desde as pequenas animações de vídeo usadas nas telecomunicações móveis até à transmissão de televisão em alta definição. A codificação, utilizando MPEG-4 AVC/H.264, é sem dúvida melhor do que a que se encontra a ser utilizada actualmente na indústria televisiva (MPEG-2). O MPEG-4 AVC/H.264 permite maiores taxas de compressão sem grandes perdas na qualidade gráfica, em detrimento do MPEG-2. Permite ainda uma maior rentabilidade da largura de banda da rede contratada. Por outro lado o MPEG-4 AVC/H.264 necessita de uma maior quantidade de recursos computacionais disponíveis para o processamento da informação gráfica e de um maior investimento inicial em sistemas computacionais mais evoluídos. Tendo em conta todas as melhorias alcançadas, pode-se afirmar que é um bom investimento.

A utilização do protocolo RTP para a transmissão de dados multimédia melhora o desempenho de todo o sistema. Esta melhoria alcança-se dada a facilidade de identificação dos serviços que se encontram a ser transmitidos no *payload* do pacote. Bastando para tal ler o PT do pacote RTP para a identificação do sistema de transmissão e ler o SSRC do mesmo para a identificação do serviço em transmissão.

Neste trabalho apresenta-se um Sistema de Conversão de Vídeo funcional, rigoroso e com baixo consumo de recursos computacionais quanto ao tratamento de serviços de vídeo. Como se pode observar pelos testes realizados com os demonstradores apresentados, o sistema é uma boa solução a considerar para clientes possuidores de sistemas de recepção mais antigos (com poucos recursos computacionais) e que pretendam aderir ao nosso serviço de transmissão. Isto porque, clientes com poucos recursos computacionais não os podem despender em processamento de informação desnecessária. Com o sistema desenvolvido, o cliente só recebe os dados relativos a uma determinada resolução espacial ou SNR. Este sistema de conversão pode ser directamente usado em redes que usem H.264 escalável onde é transmitida uma determinada qualidade ou resolução ao cliente com base na capacidade de processamento do terminal ou simplesmente porque está apenas habilitado a receber até uma determinada resolução ou qualidade.

Este trabalho apresenta ainda vantagens ao nível da segurança. Como o cliente recebe unicamente o sinal do serviço a que tem direito, este não tem a "tentação" de aceder ilegalmente a serviços que não contratou/pagou.

Como dito anteriormente, a solução desenvolvida apresenta soluções quanto à protecção dos conteúdos que se encontram a ser transmitidos na rede. O enviar para o

cliente apenas os conteúdos a que este tem acesso quando contrata o nosso serviço é outra vantagem neste sistema de Conversão de Vídeo. Estas soluções representam um avanço significativo mas não o suficiente para a total segurança dos serviços. Assim, recomenda-se que seja implementado um sistema de encriptação na carga paga do pacote RTP, dificultando desta forma ainda mais o acesso aos conteúdos.

Este projecto pode e deve continuar a ser desenvolvido posteriormente, de maneira a torna-lo cada vez mais robusto e versátil, dado que, actualmente só permite ter um único cliente ligado ao sistema de conversão.

Assim, o sistema criado no decorrer deste trabalho científico funciona com elevado desempenho em sistemas de baixos recursos computacionais, como é o caso do MINI PC utilizado nos demonstradores desenvolvidos. Pode-se afirmar que todos os objectivos proposto no início deste trabalho foram alcançados com sucesso, como era espectável.

Bibliografia

[1] Beatriz Sousa Santos, "Computação Visual", 2007-2008, http://www.ieeta.pt/~bss/aulas/Cor-11-07.pdf

[2] Rigor, Laboratório Óptico, "Anatomia do Olho Humano"- http://www.laboratoriorigor.com.br/anatomia.html

[3] MITOpenCourseWare, "Biomedical Optics", http://ocw.mit.edu/OcwWeb/Health-Sciences-and-Technology/HST-569Fall-2006/CourseHome/index.htm

[4] Poynton Charles, "Digital Video and HDTV – Algorithms and Interfaces", Morgan Kaufmann Publishers, 2003, ISBN 1-55860-792-7

[5] Richardson, Iain E.G., "H.264 and MPEG-4 Video Compression", John Willey and Sons, 2003, ISBN 0-470-84837-5

[6] Wertheimer, Max, "Experimental Studies on the Seeing of Motion", 1912

[7] Answers Corporation, http://www.answers.com/topic/rgb?cat=technology

[8] Russ, John C., "Image Processing Handbook", fifth edition, CRC Press, 2007, ISBN 0-8493-7254-2

[9] Telecommunication Standardization Sector, Recommendation ITU-R BT.601-5 – "Studio Encoding Parameters of Digital Television for Standard 4:3 wide-screen 16:9 Aspect Ratios", ITU-T, 1995

[10] Smith, Alvy Ray "Color Gamut Transform Pairs", Computer Graphics Lab, 1978

[11] Win Topo, http://homepage.ntlworld.com/heatons/softsoft/wintopo/help/html/threshold-hsv.htm

[12] Computer Science and Software Engineering http://www.csse.uwa.edu.au/~wongt/matlab.html

[13] António Navarro, "Apontamentos Teóricos de Processamento de Sinais de Vídeo - 2006/2007"

[14] Telecommunication Standardization Sector, Recommendation ITU-R BT.500-11 – "Methodology for the subjective assessment of the quality of television pictures, 2002

[15] Video Quality Experts Group – http://www.vqeg.org/

[16] ISO/IEC 14495-1:2000 Information – lossless and near-lossless compression of continuous-tone still images: Baseline, (JPEG-LS)

[17] K. R. Rao e P.C. Yip, "The Transform and Data Compression Handbook", CRC Press, 2000, ISBN 0-8493-3692-9

[18] Keith Jack, "Video Demystified: A Handbook for the Digital Engineer", Fourth Edition, Newnes, 2005, ISBN 0-7506-7822-4

[19] Huffman, D.A., "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the IRE, Volume 40, Issue 9, Setembro de 1952, Páginas: 1098 – 1101

[20] International Organization for Standardisation, ISO/IEC 14496- Parte 10 JTC1/SC29/WG11, "Coding of moving Pictures and Audio", Março, 2003

[21] Joint Video Team (JVT) de ISO/IEC MPEG & ITU-T VCEG, "Coding of audio-visual objects", Dezembro, 2003

[22] International Organization for Standardisation (ITU-T Recommendation H.264), "Advanced video coding for generic audiovisual services", Março, 2005

[23] C.S Manjunath, Philippe Salembier e Thomas Sikora; "Introduction to MPEG-7: Multimedia Contend Description Interface", John Wiley and Sons, 2002, ISBN 0-4714-8678-7

[24] Ian S. Burnett, Fernando Pereira, Rik Van de Walle e Rob Koenn, "The MPEG-21 Book", John Wiley and Sons, 2006, ISBN 0-470-01011-8

[25] Terms and Reference, MPEG home page, http://www.chiariglione.org/mpeg/

[26] Thomas Wiegand, Gary J, Sullivan, Gisle Bjontegaard e Ajay Luthra, "Overview of the H.264 Video Coding Standard", IEEE Transactions on Circuits and Systems for Video Technology, Volume 13, Número 7, Julho, 2003

[27] Elecard StreamEye Application - http://www.elecard.com

[28] Detlev Marpe, Heiko Schwarz e Thomas Weigand, "Context-Based Adaptive Binary Arithmetic Coding in the H.264 Video Compression Standard", IEEE Transactions on Circuits and Systems for Video Technology, Volume 13, Número 7, Julho, 2003

[29] International Organization for Standardisation Coding of Moving Pictures and Audio, ISO/IEC N54467, "Amendment 3: Transport of AVC video data over ITU-T Rec H.222.0 ISO/IEC 12818-1 streams", Fevereiro, 2003

[30] S.Wenger, M.M. Hannuksela, T. Stockhammar, M. Westerlund, D.Singer, "RTP Payload Formar for H.264", Network Working Group, RFC3984, Fevereiro, 2005

[31] H.Schulrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", Network Working Group, RFC3550, Julho, 2003

[32] Addison Wesley, "RTP: Audio and Video for the Internet", Colin Perkins, 2003, ISBN 0-672-32249-8

[33] M. Handley e V. Jacobson, "SDP: Session Description Protocol", Internet Engineering Task Force, RFC 2327, Abril, 1998

[34] GNU Operating System - http://www.gnu.org/

[35] Wireshark - http://www.wireshark.org/about.html

[36] Hexedit – http://www.catch22.net/software/hexedit.asp

[37] SUIT – Scalable, Ultra-fast and Interoperable Interactive Television – http://suit.av.it.pt

Anexos

- Anexo 1: Exemplos de Decomposição em espaço de cor HSV e HSL
- Anexo 2: Entrelaçamento no formato de amostragem 4:2:0
- Anexo 3: Exemplo de PSNR
- Anexo 4: Compensação de Movimento – A importância do tamanho do bloco
- Anexo 5: Decomposições base da Transformada Discreta de Co-Senos
- Anexo 6: Transformada Inversa de Co-Senos - IDCT
- Anexo 7: Lançamento dos módulos do Demonstrador 1
- Anexo 8: Lançamento dos módulos do Demonstrador 2

Anexo 1

Conteúdo:

i. Exemplo de conversão de uma imagem em HSL

ii. Exemplo de conversão de uma imagem em HSV

i.

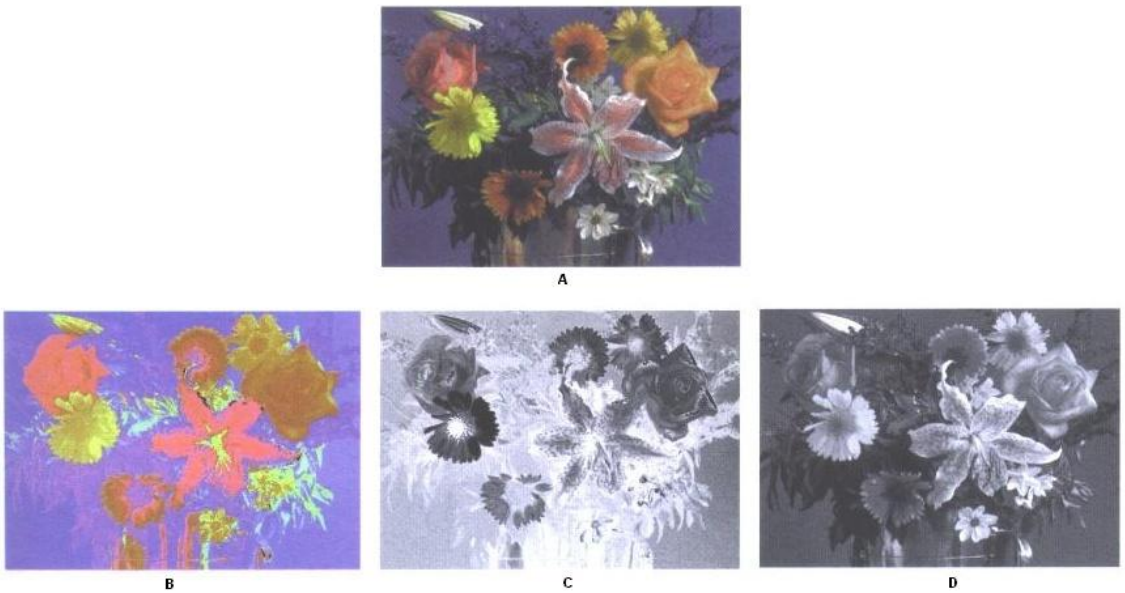

Figura 80 – Fotografia a cores de jarro de flores (fig.A) e respectiva divisão em componentes HSL (figs.B, C e D) (adaptada de [8])

Da figura anterior pode-se verificar o resultado da divisão de uma imagem real (Figura 80-A) em componentes HSL, a que correspondem a Figura 80-B à componente H, a componente S à Figura 80-C e à componente L a Figura 80-D [8].

ii.

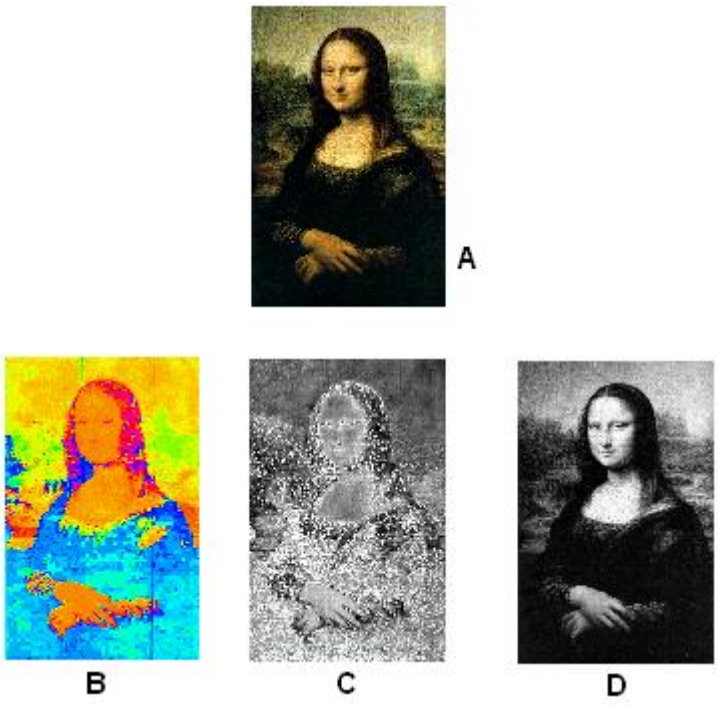

Figura 81 - Fotografia de "La Gioconda", de Leonardo Da Vinci e sua decomposição em componentes HSV (adaptada de [12])

A Figura anterior apresenta na imagem A, a imagem real, e nas imagens B, C e D a respectiva decomposição em componentes H, S e V [12].

Anexo 2

Conteúdo:

i. Entrelaçamento no formato de amostragem 4:2:0

i.

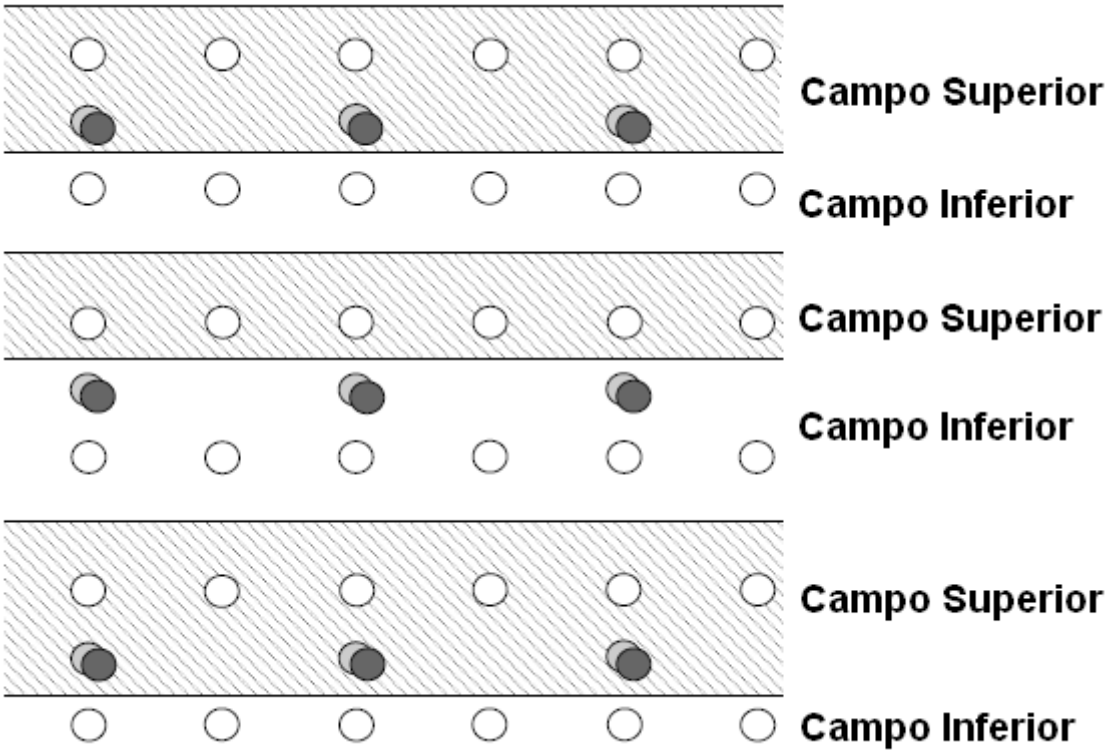

Figura 82- Alocação de amostras 4:2:0 em dois Campos, Inferior e Superior (adaptada de [5])

Na figura mostra-se o método de alocação de Y, C_b e C_r num par de campos, Superior e Inferior, adoptado no MPEG-4 Visual e no H.264. Pela figura fica claro que o número total de amostras no par de campos é o mesmo que se teria caso se tivesse uma frame em modo progressivo [5].

Anexo 3

Conteúdo:

i. Exemplo de PSNR

i.

Figura 83 - Exemplos de PSNR (adaptada de [5])

Na figura anterior é apresentada uma frame original (A), e de seguida duas imagens degradadas em relação à imagem original, (B) e (C). A imagem (B) apresenta um PSNR de 30.6 dB e a imagem (C) um PSNR de 28.3 dB, reflectindo assim que quanto menor o valor de PSNR pior é a qualidade da imagem, ou seja, melhor ou pior é o nosso codificador.

Anexo 4

Conteúdo:

i. Compensação de Movimento – A importância do tamanho do bloco

i. Neste ponto apresentam-se as imagens associadas ao estudo da importância do tamanho do bloco na compensação de movimento de uma sequência de vídeo.

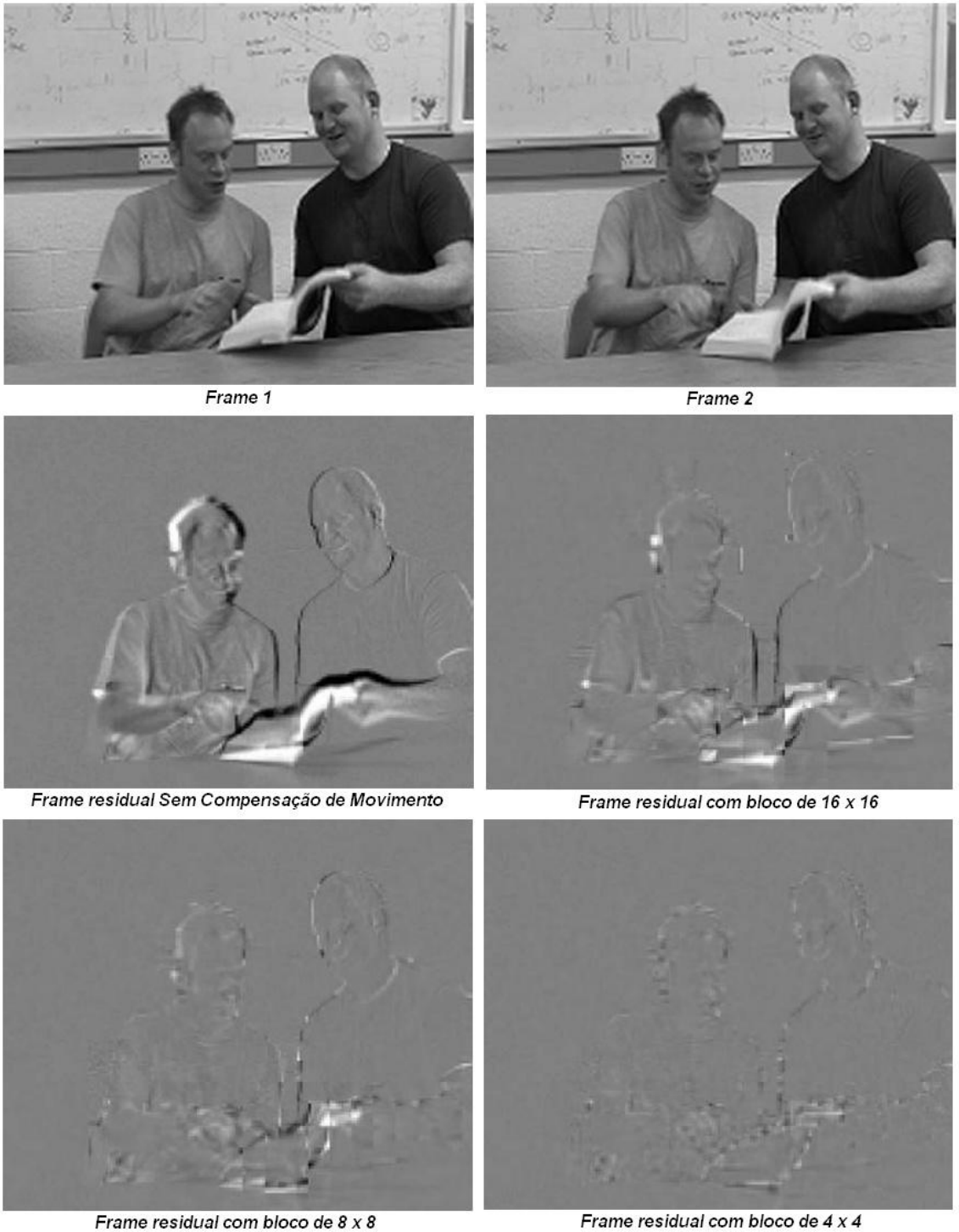

Figura 84 - Importância do tamanho do bloco na compensação de movimento (adaptada de [5])

Anexo 5

Conteúdo:

i. Decomposição Base da DCT de dimensão 4 x 4

ii. Decomposição Base da DCT de dimensão 8 x 8

i.

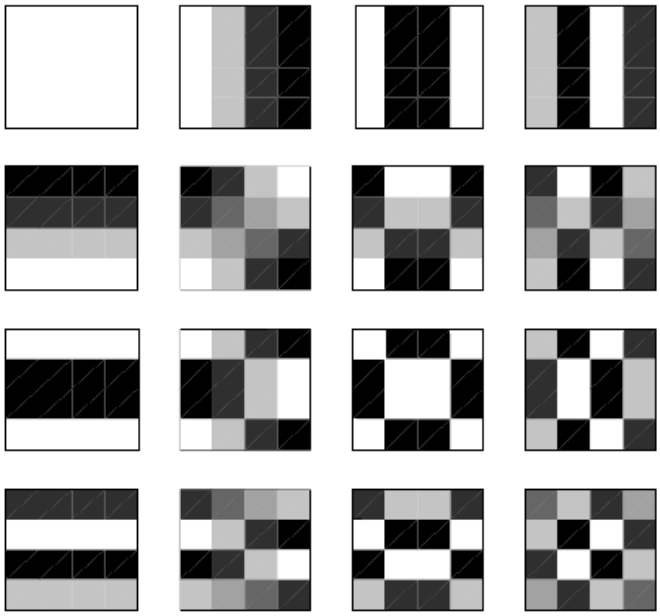

Figura 85 - Decomposição base da DCT de dimensão 4 x 4 (adaptada de [5])

ii.

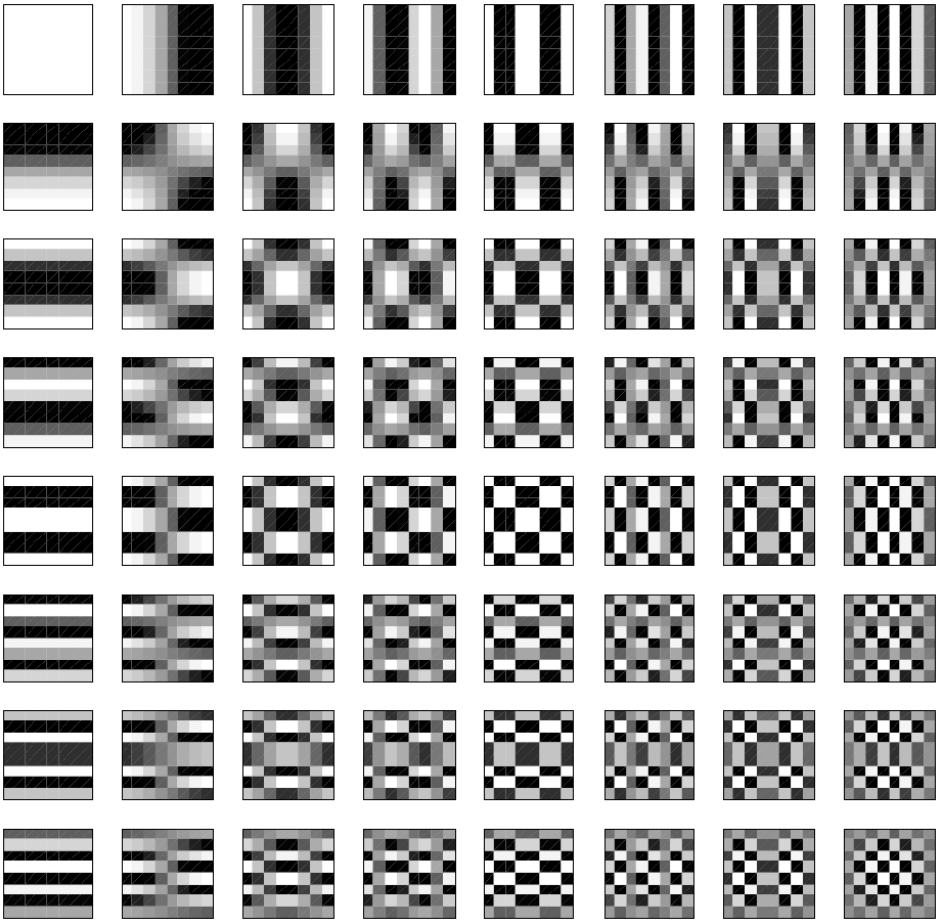

Figura 86 - Decomposição base da DCT de dimensão 8 x 8 (adaptada de [5])

As figuras representam o tratamento da transformada DCT para as dimensões 4 x 4 e 8 x 8 e as combinações verticais e horizontais das funções co-seno.

Anexo 6

Conteúdo:

i. Bloco de Imagem, zoom a bloco de dimensão 4 x 4

ii. Reconstrução de bloco de imagem por aplicação de IDCT

i.

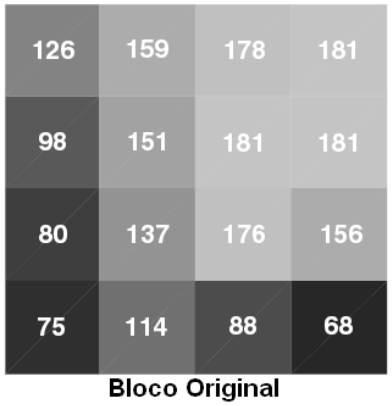

Figura 87 - Zoom a bloco de imagem de dimensão 4 x 4

ii.

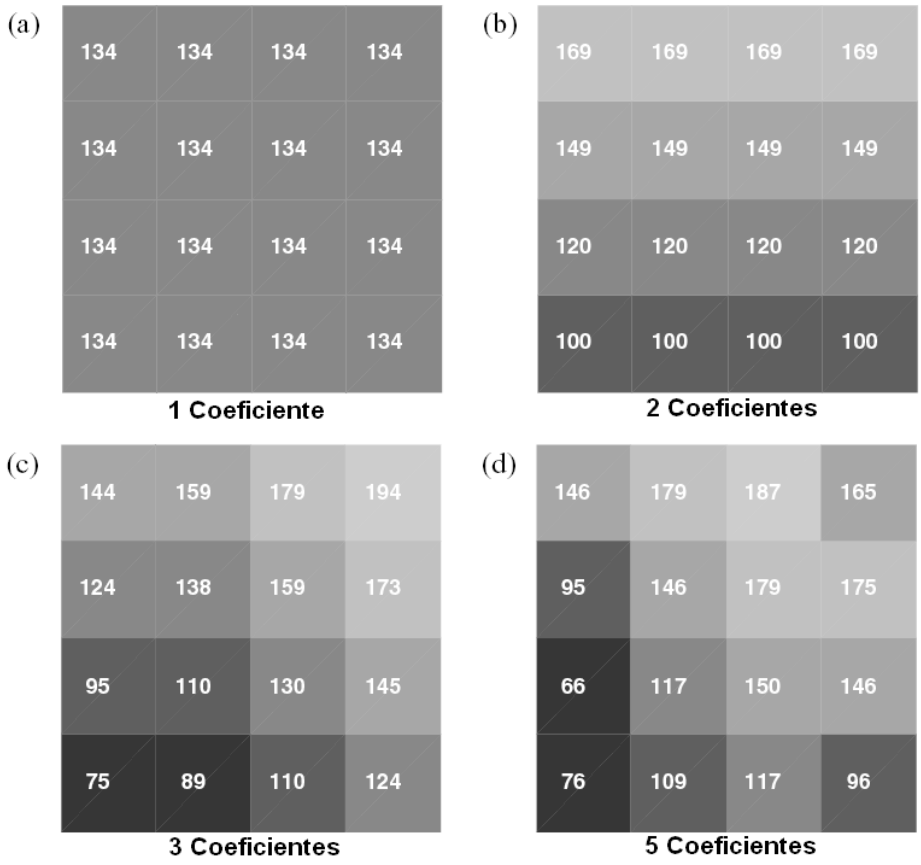

Figura 88 - Processo de reconstrução de bloco de imagem (adaptada de [5])

Da análise das Figuras anteriores podemos concluir que um bloco original retirado de uma *frame* de vídeo (i.) pode ser reconstruído, após ser transformado por uma passagem de transformada DCT, quase de forma fidedigna. Podemos verificar que à medida que o processo iterativo se dá a aproximação aos coeficientes reais do bloco da imagem é melhor. Quanto maior o número de coeficientes usados na reconstrução da imagem melhor e com maior qualidade será a reconstrução do bloco (ii.).

Anexo 7

Conteúdo:

i. Execução do módulo do Cliente do Demonstrador 1
ii. Execução do módulo do Conversor do Demonstrador 1
iii. Envio de serviço CIF no módulo de Transmissão do Demonstrador 1
iv. Envio de serviço HD no módulo de Transmissão do Demonstrador 1
v. Envio de serviço SD no módulo de Transmissão do Demonstrador 1

i.

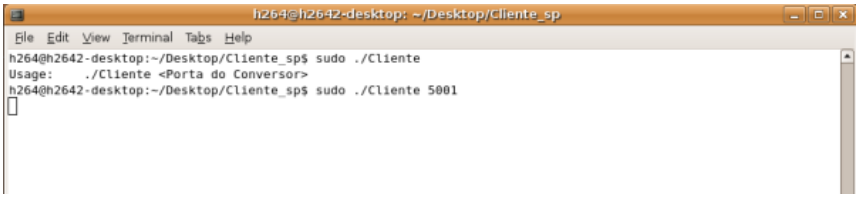

Figura 89 - Lançamento do módulo do Cliente no Demonstrador 1

ii.

Figura 90 - Lançamento do módulo do Conversor do Demonstrador 1

iii.

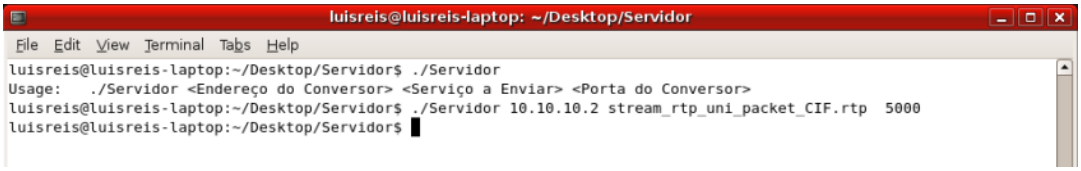

Figura 91 - Envio de serviço CIF no módulo de Transmissao de pacotes RTP, do Demonstrador 1

vi.

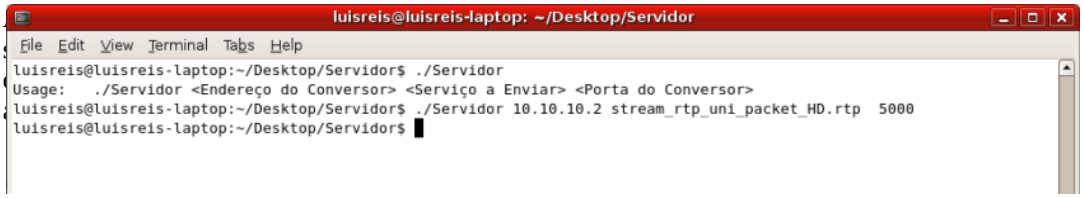

Figura 92 – Envio de serviço HD no módulo de Transmissão de pacotes RTP, do Demonstrador 1

vii.

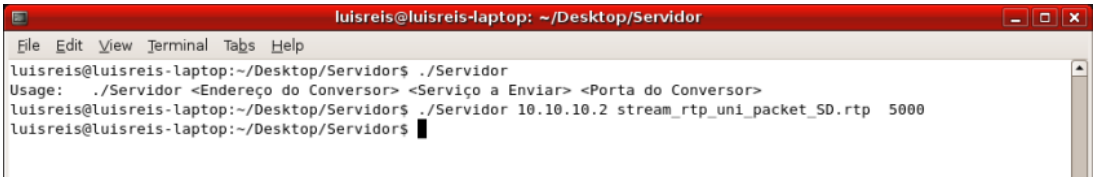

Figura 93 - Envio de serviço SD no módulo de Transmissão de pacotes RTP, do Demonstrador 1

Anexo 8

Conteúdo:

i. Lançamento do módulo do Cliente do Demonstrador 2
ii. Lançamento do módulo do Conversor do Demonstrador 2
iii. Lançamento do módulo do PLAYOUT do Demonstrador 2

i.

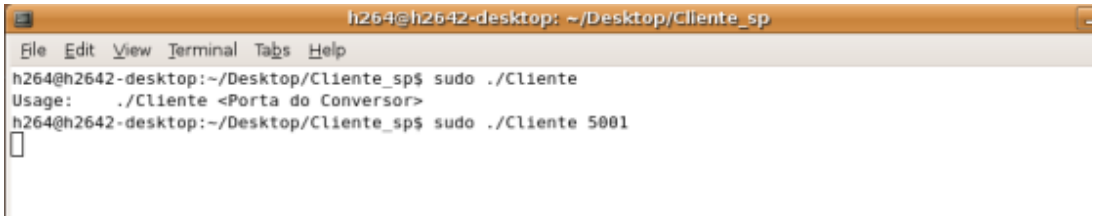

Figura 94 - Lançamento do módulo do Cliente no Demonstrador 2

ii.

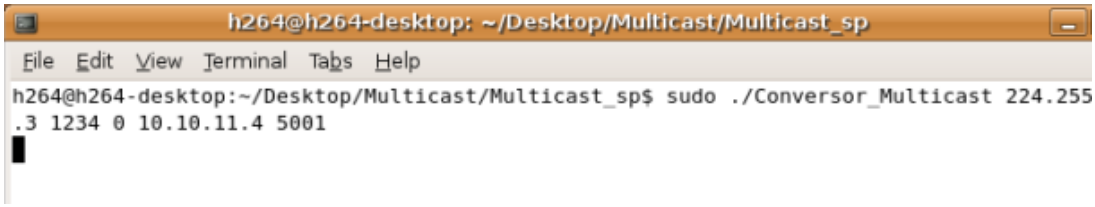

Figura 95 - Lançamento do módulo do Conversor do Demonstrador 2, para o serviço CIF

iii.

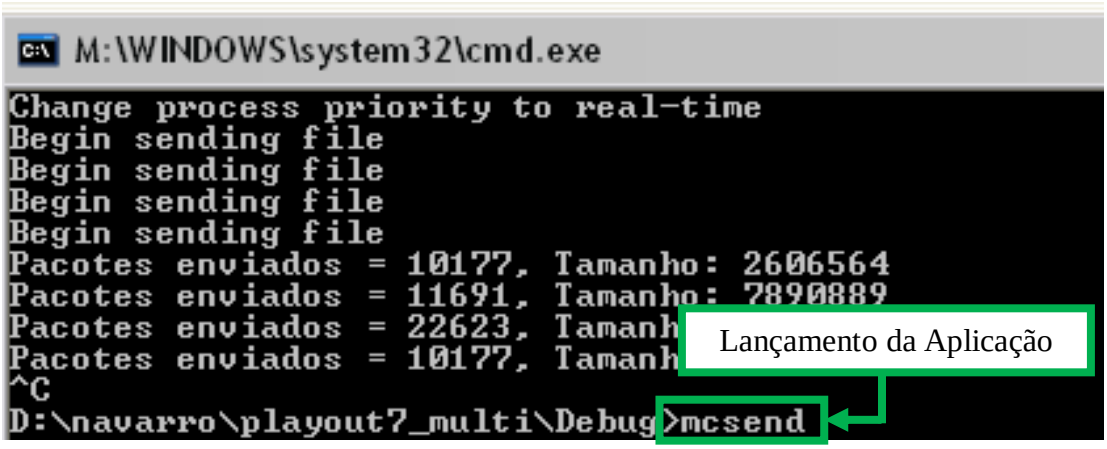

Figura 96 - Lançamento do módulo PLAYOUT do Demonstrador 2

Estrutura do CD
"Sistema de Conversão de Vídeo – Conteúdos Práticos"
Anexo a este Documento