# O padrão MPEG para compressão de Vídeo


**H. S. Nigri**
**Marcio Portes de Albuquerque**
**Marcelo Portes de Albuquerque**
**M. Taves**

**Março 2000**

# O padrão MPEG para compressão de Vídeo

# O padrão MPEG para compressão de Vídeo

## Introdução

Um vídeo não comprimido requer grande espaço de armazenamento e grande largura de banda para transmissão. Armazenar duas horas de vídeo NTSC digitalizado e não comprimido, requer 100 GB de espaço de disco. Transmitir continuamente quadros de vídeo com dimensões de 640 x 480, profundidade de cor de 24 bits, sem compressão, a 30 quadros por segundo, requer uma largura de banda de aproximadamente 200 Mb/s. Podemos notar, com os exemplos citados, que o vídeo possui grande quantidade de dados, dificultando, portanto, o seu armazenamento e a sua transmissão. Para minimizar este problema, foram criados algoritmos especiais que tiram vantagens das características do vídeo para conseguirem altas taxas de compressão.

O padrão de compressão de áudio e vídeo MPEG é um dos mais utilizados e difundidos internacionalmente, além de possuir alta aceitabilidade pelos principais aplicativos. Podemos verificar o uso do MPEG em empresas comerciais, nas instituições de ensino e pesquisa e na Internet.

Esta nota técnica apresentará uma visão geral das características comuns dos padrões de compressão de vídeo MPEG-1 e MPEG-2, em relação à codificação e decodificação.

## Padrões MPEG

Na década de 80, foi fundado um grupo de pesquisa conhecido como *Moving Picture Experts Group* ou MPEG, tendo como objetivo principal criar padrões internacionais para compressão, descompressão e representação codificada para áudio e vídeo. Como resultado do trabalho, foram criados os padrões MPEG.

O padrão MPEG-1 aprovado em 1992, é muito utilizado no armazenamento de filmes em CD-ROM com taxa de transferência de 1,5 Mb/s.

O padrão MPEG-2 aprovado em 1994, foi originalmente projetado para comprimir vídeo em sistemas de difusão *(broadcast)* a taxas de 4 a 9 Mb/s. Posteriormente, o MPEG-2 foi expandido e passou a incorporar as características do então abandonado padrão MPEG-3, ou seja, suportar a televisão de alta definição HDTV *(High Definition TV)*. Os padrões MPEG-1 e MPEG-2 são similares em seus conceitos básicos. Ambos são baseados na compensação de movimento e nas técnicas de codificação baseadas em transformadas da imagem.

O padrão MPEG-4, aprovado em 1999, foi projetado para atender a aplicações multimídia com baixas taxas de transferência, indo de 5 a 64 Kb/s. Alguns exemplos de aplicações que utilizam ou venham a utilizar o MPEG-4 são aplicações gráficas interativas, WWW, videophone e correio eletrônico multimídia. A codificação MPEG-4 é baseada em objetos. Um objeto pode ser uma imagem, um vídeo ou um áudio. Ex.: um carro em movimento, uma fotografia de um cão, um instrumento de uma orquestra ou um latido de um cão.

O padrão MPEG-7 deverá ser aprovado em 2001 e seu objetivo é especificar um conjunto de padrões descritores de informação multimídia, com a finalidade de tornar a busca deste tipo de informação na *World Wide Web* mais rápida e eficiente. Muitas aplicações serão beneficiadas com o padrão MPEG-7: bibliotecas digitais (catálogo de imagens, dicionário musical), serviços de diretório multimídia (páginas amarelas), seleção de mídia de difusão (canais de rádio e TV), jornalismo (pesquisa de noticiários antigos), entre outras.

## A Importância da Compressão

Um vídeo é uma seqüência de quadros. Cada um desses quadros é uma imagem estática. Para armazenar uma imagem com as dimensões de 640x480 (VGA) e com uma profundidade de cor de 24 bits (codificação RGB com uma resolução de 8 bits por componente), precisamos de aproximadamente 900 KBytes:
640 x 480 = 307.200 pixels
307.200 pixels x 24 bits/pixel = 7.372.800 bits ou
7.372.800 bits / 8 bits / 1.024 = 900 KB

Geralmente, um vídeo apresenta 30 quadros por segundo. A largura de banda necessária para transmitir um fluxo contínuo desses quadros de vídeo a 30 quadros por segundo, é de, aproximadamente, 200 Mbits/s:
900 KB/imagem x 30 imagens/s = 27.000 KB/s ou
27.000 KB/s x 8 bits x 1.024 = 200 Mbits/s

Como exemplo, um CD-ROM, que tem uma capacidade de armazenamento de 640 MBytes, consegue gravar 72 minutos de áudio descomprimido. No entanto, este mesmo CD-ROM apenas conseguiria armazenar 30 segundos de vídeo digital descomprimido com qualidade de TV. Verificamos, portanto, que um vídeo digital descomprimido, com duração de 90 minutos, exige aproximadamente 120 GBytes de espaço de armazenamento em disco.

Os exemplos acima servem para demonstrar a grande importância e necessidade de se comprimir as informações de vídeo.

## Modos de compressão

A compressão e descompressão de vídeo pode ser realizada por software ou por hardware. A solução por software é utilizada pela maioria dos usuários, pois, apesar de mais lenta, é uma implementação barata. Estes softwares exigem muito da CPU. Por este motivo, foram criadas instruções e características na arquitetura das CPU's que facilitam o uso de aplicativos multimídia. É o caso do MMX da Intel, *Sun's Visual Instruction Set* da Sun e HP's MAX-2 da HP.

O hardware que realiza esta operação é composto de um chip VLSI dedicado a compressão de vídeo, uma memória DRAM e um microprocessador de baixo custo.

A compressão pode ser realizada sem perdas *(lossless compression)*, onde a informação é fielmente recuperada após o processo de descompressão, isto é, o fluxo de bits descomprimido é idêntico ao fluxo de bits original. Há também a compressão com perdas *(lossy compression)*. Como o olho humano não é capaz de notar pequenas mudanças de um quadro de vídeo para outro, esta técnica não codifica todos os detalhes em um vídeo. Alguns desses detalhes são perdidos. É possível obter altas taxas de compressão quando esse modo é utilizado.

## Técnicas de compressão

Num vídeo, podemos observar que há grande quantidade de informação se repetindo nos quadros consecutivos. Se uma árvore é mostrada durante um segundo, então 30 quadros vão conter essa mesma árvore. A informação do primeiro quadro pode ser usada na compressão, e os quadros posteriores da seqüência de vídeo poderão ser baseados nos quadros anteriores. Por exemplo, quadros posteriores poderão ter uma informação como "mova essa parte da árvore para esse lugar" em vez de conter dados redundantes de quadro para quadro.

Nas imagens, cada cor pode ser representada como uma combinação de vermelho, verde e azul. Esta representação é chamada de RGB *(red, green, blue)*. Entretanto, essa representação não é adequada para a compressão, pois não leva em conta a percepção do olho humano. As informações de brilho e de cor são tratadas de forma diferente pelo sistema visual humano. Somos mais sensíveis às mudanças no brilho do que as cores. Devido a isso, um componente especial é usado para representar informações de brilho. Este componente é chamado de luminância. A representação de cor RGB é então convertido para o espaço de cor YCbCr, onde Y é o sinal luminância, Cb é o sinal crominância diferença azul e o Cr é o sinal

crominância diferença vermelho. O espaço de cor utilizado pelo MPEG é o YCbCr, pois ele consegue obter taxas maiores de compressão quando comparado com o espaço de cor RGB.

A passagem de uma imagem RGB para uma imagem YCbCr é feita da seguinte forma:

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

$Cb = -0.1687R - 0.3313G + 0.5B + 128$

$Cr = 0.5R - 0.4187G - 0.0813B + 128$

Para entendermos melhor, vamos ao exemplo: Suponha que tenhamos um pixel com a cor marrom, onde os valores dos componentes RGB são os seguintes:

R=120 G=60 B=30

Após convertermos esse pixel para a representação YCbCr, teremos:

$Y = 0.299R + 0.587G + 0.114B = 0.299 * 120 + 0.587 * 60 + 0.114 * 30 = 74.52$

$Cb = -0.1687R - 0.3313G + 0.5B + 128 = - 0.1687*120 - 0.3313*60 + 0.5*30 + 128 = 102.88$

$Cr = 0.5R - 0.4187G - 0.0813B + 128 = 0.5 * 120 - 0.4187 * 60 - 0.0813 * 30 + 128 = 160.44$

Portanto, a cor marrom na representação RGB, onde R=120, G=60 e B=30, possui na representação YCbCr, os valores Y=74.52, Cb=102.88 e Cr=160.44.

Os algoritmos de compressão costumam dividir os quadros de uma seqüência de vídeo em blocos e macroblocos. Em vez de analisarem pixel por pixel desses quadros, analisarão blocos por blocos. Um bloco consiste em um grupo de 8 pixels horizontais por 8 pixels verticais, enquanto que um macrobloco consiste em um conjunto de 16 x 16 pixels.

Um macrobloco pode ser representado de diferentes maneiras quando nos referimos ao espaço YCbCr.

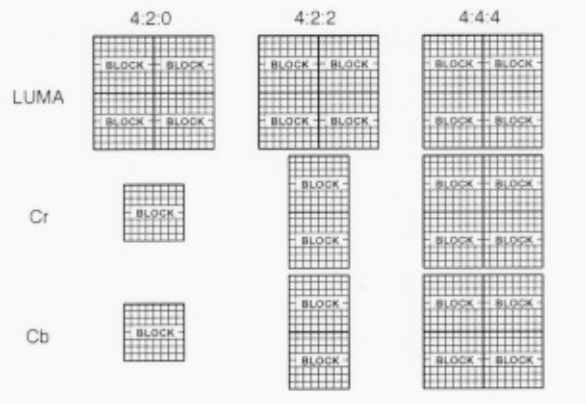


Figura 1

A figura 1 mostra três formatos conhecidos como 4:4:4, 4:2:2 e 4:2:0. No formato 4:4:4, para cada 4 blocos de Y, temos 4 de Cb e 4 de Cr. No formato 4:2:2, para cada 4 blocos de Y, temos 2 de Cb e 2 de Cr. O formato 4:2:0 contém apenas um quarto da informação de crominância, ou seja, para cada 4 blocos de Y, temos apenas 1 de Cb e 1 de Cr. Devido a maneira eficiente de representar a luminância e a crominância, o formato 4:2:0 permite uma redução imediata de informação de 12 para 6 blocos. Apesar do MPEG-2 ter provisão para lidar com o formato 4:4:4 para aplicações profissionais, a maioria das aplicações usam o formato 4:2:0.

Finalmente, é importante definirmos dois tipos de redundância exploradas pelos algoritmos de compressão de vídeo MPEG: a redundância espacial, que é a redundância contida dentro de um mesmo quadro de vídeo, e a redundância temporal, que é a redundância existente na seqüência de quadros.

## Técnicas de codificação

### Técnica de Codificação Intra-quadro

O termo “codificação intra-quadro” se refere a codificação realizada dentro de um mesmo quadro, não havendo relação com os outros quadros da seqüência de vídeo.

O diagrama em blocos da figura 2 apresenta a codificação intra-quadro. Podemos notar os seguintes blocos: o *Video Filter* (filtro de vídeo), o DCT- *Discrete Cosine Transform* (transformada discreta do coseno, ou seja, a transformada da imagem utilizada), o *Quantizer* (quantizador dos coeficientes DCT), o RLE - *Run Length amplitude/Variable Length Coder* (codificador de largura variável), o *Bitstream Buffer* (buffer de vídeo) e o B*it-rate control* (controle de taxa de bits). Esses blocos são descritos a seguir.

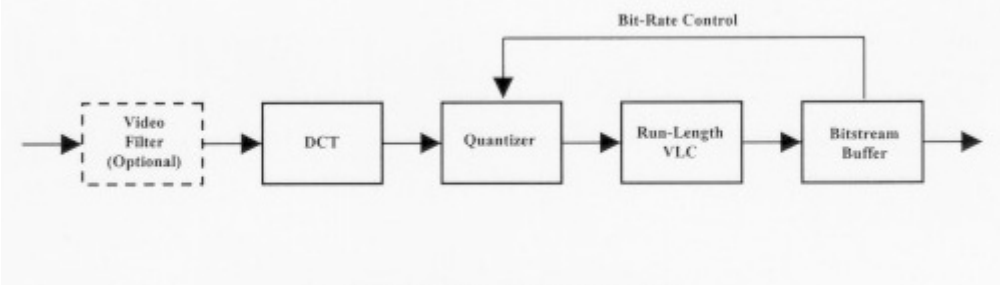


Figura 2

### *1) Filtro de vídeo*

Para que não haja distorções nas cores ao se utilizar o formato 4:2:0, é preciso filtrar os sinais relativos a crominância. Por este motivo, é recomendável a utilização do filtro de vídeo.

### *2) Transformada Discreta do Coseno (DCT)*

A codificação baseada na transformada é feita da seguinte forma: a informação original sofre uma transformação matemática do domínio espacial ou temporal para um domínio abstrato, que é mais adequado para a compressão. O processo é reversível, isto é, a aplicação da transformada inversa reconstitui a informação original.

Existem várias transformações matemáticas (Coseno, Hadamar, Haar, Fourier, etc.). Como sabemos, qualquer função matemática pode sofrer uma transformação para outro domínio de variáveis e valores. Um exemplo muito conhecido é a transformada de Fourier: uma medida que varia no tempo *f(t)* pode ser transformada pelo mecanismo de Fourier numa função *g(v)*. Esta nova função proporciona a amplitude (ou coeficiente) "*g*" das freqüências "*v*" que compõem a função inicial. Por isso, dizemos que *g(v)* é a distribuição espectral da função *f(t)*.

As transformações matemáticas utilizadas com maior freqüência para a compressão de imagens e vídeos são a Transformada Discreta do Coseno ou DCT (*Discrete Cosine Transform*) e sua transformada inversa, a IDCT *(Inverse DCT)*, pois são as mais adequadas para este tipo de dados. A DCT e a IDCT são apresentadas na equação 1 e 2 abaixo.

A DCT transforma o sinal original para o domínio da freqüência. Após efetuar a DCT, os coeficientes mais significativos podem ser codificados com maior precisão, e os menos significativos, com menor precisão. É igualmente possível ignorar alguns coeficientes. A transformação é reversível e pode ser usada no modo de compressão sem perdas.

$$F(\mu, \nu) = \frac{1}{4} C(\mu)C(\nu)\sum_{x=0}^{7}\sum_{y=0}^{7} f(x, y) \cos\left[\frac{(2x + 1)\mu\pi}{16}\right] \cos\left[\frac{(2y + 1)\nu\pi}{16}\right]$$

Equação 1: *Transformada Discreta do Coseno*

$$f(x, y) = \frac{1}{4} \sum_{\mu=0}^{7}\sum_{\nu=0}^{7} C(\mu)C(\nu)F(\mu, \nu) \cos\left[\frac{(2x + 1)\mu\pi}{16}\right] \cos\left[\frac{(2y + 1)\nu\pi}{16}\right]$$

Equação 2: *Transformada Inversa Discreta do Coseno*

A DCT apresentada na equação 1, é aplicada a cada um dos blocos 8 x 8 (64 pixels) de um quadro que será comprimido.

As coordenadas $x$ e $y$, e $\mu$ e $\nu$, indicam uma das oito posições horizontais ($y$ e $\nu$) e verticais ($x$ e $\mu$) possíveis para os 64 pixels de um bloco, onde:

- $x$ e $y$ são as coordenadas da matriz espacial (original).
- $\mu$ e $\nu$ são as coordenadas da nova matriz gerada pela Transformada Discreta do Coseno.
- $f(x,y)$ é o valor do pixel na posição $(x,y)$.
- $F(\mu,\nu)$ é o valor da DCT na posição $(\mu,\nu)$.
- $C(\mu)$ e $C(\nu) = 1/\sqrt{2}$ ( para $\mu$ ou $\nu = 0$) ou $C(\mu)$ e $C(\nu) = 1$ ( para $\mu$ e $\nu \neq 0$)

A figura 3a é um exemplo de um bloco 8x8 qualquer de uma imagem a ser comprimida. Cada um dos valores assinalados na figura 3a representa o valor da intensidade luminosa (Y) de um pixel da imagem. A figura 3b é o mesmo bloco da figura 3a após a aplicação da DCT.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 64,24 | 47,74 | 42,63 | 34,95 | 30,81 | 25,87 | 21,82 | 19,78 |
| 45,12 | 42,20 | 37,01 | 31,34 | 27,70 | 23,13 | 19,93 | 17,53 |
| 38,55 | 34,55 | 30,83 | 28,21 | 23,04 | 19,23 | 16,91 | 15,89 |
| 28,61 | 28,22 | 26,30 | 22,78 | 18,86 | 16,10 | 15,08 | 14,30 |
| 23,59 | 22,69 | 21,11 | 18,71 | 15,72 | 12,93 | 13,01 | 11,47 |
| 18,15 | 16,86 | 15,99 | 14,62 | 12,87 | 11,74 | 11,13 | 10,98 |
| 15,83 | 14,04 | 12,44 | 11,29 | 10,37 | 10,89 | 11,00 | 10,46 |
| 13,36 | 10,67 | 10,09 | 9,41 | 9,88 | 10,67 | 10,99 | 11,46 |

Figura 3a Bloco de pixels original

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 168 | 45 | 7 | 3 | 2 | 2 | 1 | 1 |
| 67 | 32 | 3 | 3 | 2 | 1 | 1 | 1 |
| 12 | 5 | 5 | 5 | 2 | 1 | 2 | 1 |
| 5 | 5 | 2 | 2 | 1 | 1 | 1 | 1 |
| 3 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 2 | 1 | 1 | 0 |
| 1 | 2 | 1 | 1 | 1 | 1 | 0 | 0 |

Figura 3b Bloco após a DCT

Vale ressaltar que, na representação espectral de imagens, as freqüências descrevem a rapidez com que as cores e a luminância se alteram em uma direção.

### *3) Quantização dos coeficientes DCT*

Os 64 coeficientes produzidos pela DCT são, então, quantizados. Esse processo de quantização é realizado dividindo cada coeficiente do bloco transformado (fig. 3b) pelo valor correspondente numa matriz chamada de matriz de quantização (fig. 4a). O resultado desta operação é a matriz quantizada (fig. 4b).

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 4 | 8 | 16 |
| 1 | 1 | 1 | 1 | 4 | 4 | 8 | 16 |
| 1 | 1 | 1 | 2 | 4 | 4 | 8 | 16 |
| 2 | 8 | 8 | 8 | 8 | 16 | 16 | 16 |
| 4 | 8 | 8 | 8 | 8 | 16 | 16 | 32 |
| 4 | 8 | 8 | 8 | 16 | 16 | 16 | 32 |
| 4 | 8 | 8 | 8 | 16 | 16 | 32 | 32 |
| 8 | 8 | 8 | 16 | 16 | 32 | 32 | 64 |

Figura 4a Matriz de Quantização

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 168 | 45 | 7 | 3 | 2 | 0 | 0 | 0 |
| 67 | 32 | 3 | 3 | 0 | 0 | 0 | 0 |
| 12 | 5 | 5 | 2 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figura 4b Bloco Quantizado

Após sofrerem a transformada, os coeficientes geralmente tornam-se pequenos e após a quantização, nulos. Nesta conversão existe uma perda de informação devido ao processo de truncagem obtido na divisão de 2 números inteiros.

De uma maneira eficiente, podemos economizar espaço na representação da grande quantidade de coeficientes DCT com valor zero (fig. 4b), reorganizando-os e aplicando o R*un-Length Encoding* - RLE (que será descrita no próximo tópico).

A reorganização é feita da seguinte maneira: como a maior parte dos coeficientes DCT diferentes de zero estarão concentrados na parte superior esquerda da matriz, um processo de varredura em *zig-zag* é aplicado, como visto na figura 5 abaixo. Esse processo agrupa, de forma seqüênciada, uma grande quantidade de zeros.

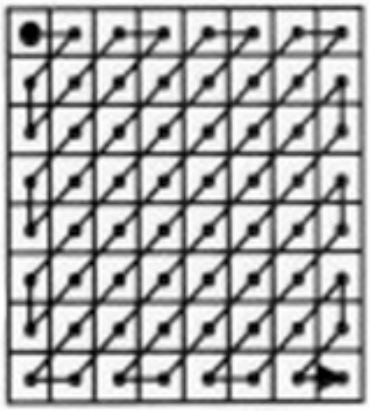

Figura 5 Varredura em zig-zag

Para demonstrar a eficiência da DCT, uma série de figuras é apresentada.


Figura 6


Figura 7

A figura 6 mostra uma imagem monocromática de 8 bits, onde uma operação DCT 8x8 foi aplicada em todos os blocos da imagem. Todos os coeficientes gerados à partir dessa operação foram mantidos. Em seguida a transformada inversa IDCT foi aplicada reconstruindo a imagem.

A figura 7 é a mesma imagem com somente 10 coeficientes DCT mantidos. Os outros 54 coeficientes DCT foram zerados. Quando a IDCT é aplicada e a imagem é reconstruída, percebemos que a qualidade é praticamente mantida, comparada com a figura 6, que foi reconstruída usando todos os 64 coeficientes da DCT.


Figura 8


Figura 9

A figura 8 mantêm apenas 6 coeficientes. Existe alguma degradação da imagem, mas mesmo assim, a qualidade continua aceitável.

A figura 9 mantêm apenas 3 coeficientes no interior de um bloco. A essa altura, observamos que a imagem começa a ficar muito quadriculada, principalmente nas bordas dos objetos onde temos o melhor contraste. Porém este efeito de "quadriculação" existirá em toda a imagem.

Figura 10

A figura 10 ilustra o caso extremo em que somente o componente DC é mantido (o coeficiente superior esquerdo da matriz), ou seja, apenas 1 dos 64 coeficientes originais foi mantido. Embora a imagem tenha ficado bem quadriculada, ela ainda é reconhecível.

### *4) RLE (***Run-Length Encoding***)*

Qualquer seqüência de caracteres repetidos pode ser substituída por uma forma abreviada. O funcionamento da técnica RLE consiste em substituir uma série de (**n)** caracteres (**c**) sucessivos pelo próprio caractere (**c**) seguido por um caractere especial (*flag*) que, por sua vez, é seguido pelo número (**n**) de ocorrência do caracter repetido. Este conjunto de três caracteres que substitui a seqüência repetida chama-se *token.* Ex.: Suponhamos que temos a seguinte *string* de caracteres: A B C C C C C C A B C A B C. Utilizando a codificação RLE podemos comprimir a sub-string de C's utilizando uma *flag* qualquer. A forma comprimida poderia assumir a seguinte forma : A B !6C A B C A B C.

Uma conclusão que se pode tirar imediatamente, é que este método não deve ser utilizado nos casos em que um caracter surge repetido apenas duas vezes. A substituição deve apenas ocorrer se o número de ocorrência do mesmo caracter for igual ou superior a quatro.

### *Codificação de Huffman*

A codificação de Huffman é utilizada quando se pretende conseguir uma melhor compactação. Neste tipo de codificação, alguns coeficientes (os mais freqüentes) são codificados com menos bits do que outros.

O método de codificação de Huffman calcula a freqüência de ocorrências de cada caracter para uma dada porção do fluxo de dados, determina o número mínimo de bits para cada caracter a partir de uma tabela de freqüência de ocorrências, e atribui um código ideal, de acordo com esse cálculo. Os códigos atribuídos são armazenados numa tabela (*code book*). Note que os códigos de Huffman são códigos de comprimento variável e sua representação permite a criação de uma seqüência única, necessária para a separação dos valores no momento de sua decodificação. Os códigos de menor comprimento são atribuídos aos caracteres que ocorrem com maior freqüência.

Exemplo: Desejamos codificar uma mensagem de 100.000 caracteres, que consiste das seguintes letras: A, B, C, D, E e F (vide tabela abaixo). Se utilizássemos um código de comprimento fixo, necessitaríamos de 3 bits por caracter. Utilizando a codificação de Huffman (código de comprimento variável), o código de maior freqüência (A) utiliza 1 bit, enquanto o código de menor freqüência (F) utiliza 4 bits.

| | **A** | **B** | **C** | **D** | **E** | **F** |
|---|---|---|---|---|---|---|
| Freqüência de ocorrência | 45.000 | 13.000 | 12.000 | 16.000 | 9.000 | 5.000 |
| Código ASCII (7 bits) | 1100001 | 1100010 | 1100011 | 1100100 | 1100101 | 1100110 |
| Código Comprimento fixo | 000 | 001 | 010 | 011 | 100 | 101 |
| Código de Huffman | 0 | 101 | 100 | 111 | 1101 | 1100 |

A tabela abaixo apresenta o número de bits utilizado por cada código para representar a mensagem acima.

| **Código** | **Número de bits** |
|---|---|
| ASCII | 700.000 |
| Comprimento fixo | 300.000 |
| Huffman | 224.000 |

Considere o bloco quantizado da figura 11 abaixo. Após a aplicação da varredura em zig-zag (figura 5), que tenderá a maximizar a probabilidade de se achar uma grande seqüência de zeros, teremos o seguinte fluxo de dados: 8, 4, 4, 2, 2, 2, 1, 1, 1, 1, 12 zeros, 1, 41 zeros.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 8 | 4 | 2 | 1 | 0 | 0 | 0 | 0 |
| 4 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figura 11

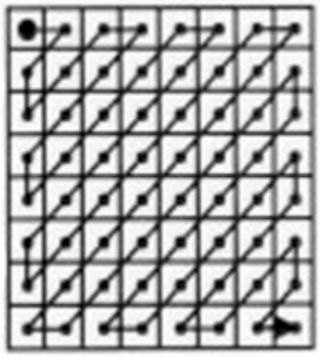

Figura 5

Essa seqüência é mostrada na tabela abaixo, onde: Run é a quantidade de zeros anteriores a cada coeficiente diferente de zero; amplitude é o valor de cada um dos coeficientes diferentes de zero; Código Mpeg é o valor da codificação de Huffman; EOB *(End Of Block)* significa fim de bloco.

| **Run** | **Amplitude** | **Código MPEG** |
|---|---|---|
| *** | 8 (Valor DC) | 110 1000 |
| 0 | 4 | 0000 1100 |
| 0 | 4 | 0000 1100 |
| 0 | 2 | 0100 0 |
| 0 | 2 | 0100 0 |
| 0 | 2 | 0100 0 |
| 0 | 1 | 110 |
| 0 | 1 | 110 |
| 0 | 1 | 110 |
| 0 | 1 | 110 |
| 12 | 1 | 0010 0010 0 |
| EOB | EOB | 10 |

Na tabela acima, na coluna do Run, onde está marcado 12, temos 12 zeros entre o 1 e o próximo valor não nulo ( que é outro 1). Os 41 zeros restantes foram desprezados, sendo representados por 2 bits de fim de bloco). Assim, temos 61 bits para representar os 64 coeficientes deste bloco. Para representar esses mesmos 64 coeficientes (pixels) sem compressão, necessitaríamos de 512 bits, considerando que cada pixel representado sem compressão requer 8 bits (64 pixels x 8 bits ).

### *5) Buffer de vídeo e controle da taxa de transmissão de bits* (Bit Rate Control)

Para transmitir informações ou fluxo de dados comprimido, a maioria das aplicações utilizam uma taxa de transferência (*bit rate*) fixa. Ex.: No caso da HDTV essa taxa é fixada em 18 Mb/s. No entanto, as imagens comprimidas apresentam diferenças na quantidade de informação a ser codificada, e portanto, o fluxo de dados comprimido não apresentará uma taxa fixa de transferência de imagem para imagem. Para resolver o problema, é utilizado um "buffer" (que possui tamanho fixo) e um sistema de realimentação (*bit rate control*), para evitar que a quantidade de dados codificada exceda ou fique bem abaixo do tamanho do buffer.

A realimentação é feita no bloco quantizador porque a matriz de quantização pode variar de figura para figura, gerando um fluxo de dados variáveis, e, assim, subcarregar ou sobrecarregar o tamanho máximo do buffer.

## Técnica de Codificação Inter-quadro

A técnica de codificação intra-quadro discutida anteriormente, é limitada a processar sinal de vídeo espacialmente, relativa a informações contidas em um único quadro de vídeo. Entretanto, taxas maiores de compressão podem ser alcançadas se redundâncias temporais, relativas a uma seqüência de quadros, puderem ser exploradas. Isso porque em uma seqüência de quadros (em um vídeo), quadros consecutivos são muito parecidos com os quadros anteriores e posteriores ao quadro de interesse. Um exemplo é dado nas figuras abaixo.

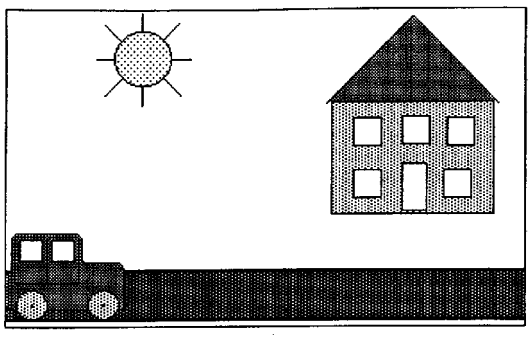

Figura 12

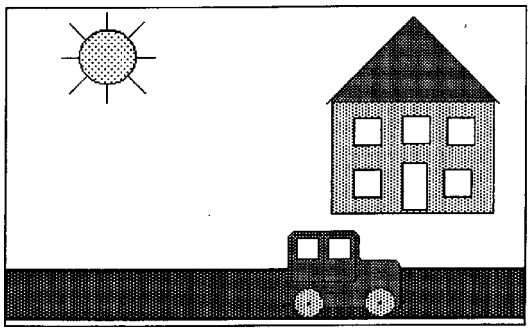

Figura 13

A redundância temporal ou inter-quadro é explorada da seguinte maneira: Alguns quadros na seqüência de vídeo serão codificados espacialmente (codificação intra-quadro) e o restante será codificado a partir de informações contidas nos quadros próximos a ele. Como exemplo, a figura 12 pode ser codificada espacialmente enquanto a figura 13 pode ser codificada com a informação de manter o cenário da figura 12 modificando apenas o carro para a direita.

A técnica de codificação temporal define três tipos de quadros: Os quadros I, P e B.

**Quadro I:** O quadro I é o primeiro quadro na seqüência de quadros codificado. Ele sofre codificação intra-quadro, ou seja, não possui referência a qualquer outro quadro da seqüência de vídeo.

**Quadro P:** A nomenclatura P vem de previsto. Isso porque um quadro P é gerado ou previsto a partir de um quadro I ou outros quadros P da seqüência de quadros do vídeo codificado. Ele contém apenas informações relativas às diferenças entre os quadros, e só pode ser gerado por um quadro I ou P imediatamente anterior a ele. Por isso ele é dito unidirecional no tempo.

Exemplo: Considere a seguinte seqüência de quadros : I P P .... O primeiro quadro P foi gerado a partir do quadro de referência I e o segundo quadro P foi gerado a partir do primeiro quadro P. O primeiro quadro P não pode ser gerado a partir do segundo P. Isso porque quadros P são unidirecionais.

**Quadro B:** O codificador tem a opção de gerar ou prever um quadro B a partir de um quadro I ou P, anterior ou posterior. Por isso, esse quadro é chamado de quadro bidirecional ou B.

Uma típica seqüência IPB é mostrada na figura abaixo. As setas representam as possíveis relações de dependência entre os quadros.

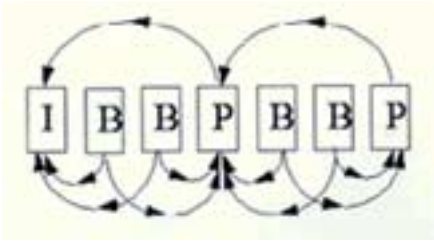


Figura 14

Em vez do codificador transmitir todos os bits que compõem os quadros da seqüência de vídeo, ele transmite apenas os bits necessários para recuperar os quadros originais.

A taxa de dados (*data rate*) típica de um quadro I é um bit por pixel. Enquanto a de um quadro P e B são, respectivamente, 0,1 bit por pixel e 0,015 bit por pixel.

Para transmitir os quadros codificados, o codificador reordena-os, de modo que os quadros que irão gerar os outros sejam transmitidos antes, fora de ordem. Ex.: Na seqüência I P B P, considerando que o quadro B seja gerado pelo quadro P posterior a ele, o codificador reordena-os, de modo que na saída do codificador teremos: I P P B.

### *Estimação de movimento*

A técnica de compressão temporal usada no vídeo MPEG é baseada na estimação de movimento. Ela consiste em comparar os pixels contidos no macrobloco (região de 16 x 16 pixels) de um quadro com os macroblocos dos quadros vizinhos. O objetivo é achar macroblocos iguais nos dois quadros, ou seja, macroblocos que contenham as mesmas informações/pixels.

A dimensão do macrobloco escolhida deu-se por proporcionar uma boa solução entre fornecer uma redução eficiente da redundância temporal e exigir um nível moderado de requisitos computacionais.

Quadro 1 Quadro 2

Figura 15

Como exemplo, vamos considerar os quadros vizinhos 1 e 2 da figura 15. Esses quadros apresentam alto grau de similaridade. Do quadro 1 para o 2, a árvore moveu-se um pouco para baixo e para a direita, enquanto que os bonecos se moveram para a direita. Na primeira fase do processo de compressão temporal por estimação de movimento, dividimos esses dois quadros em macroblocos. Em seguida, para cada macrobloco do quadro 2 determinaremos o macrobloco que melhor corresponda no quadro 1 (observe a figura 16).

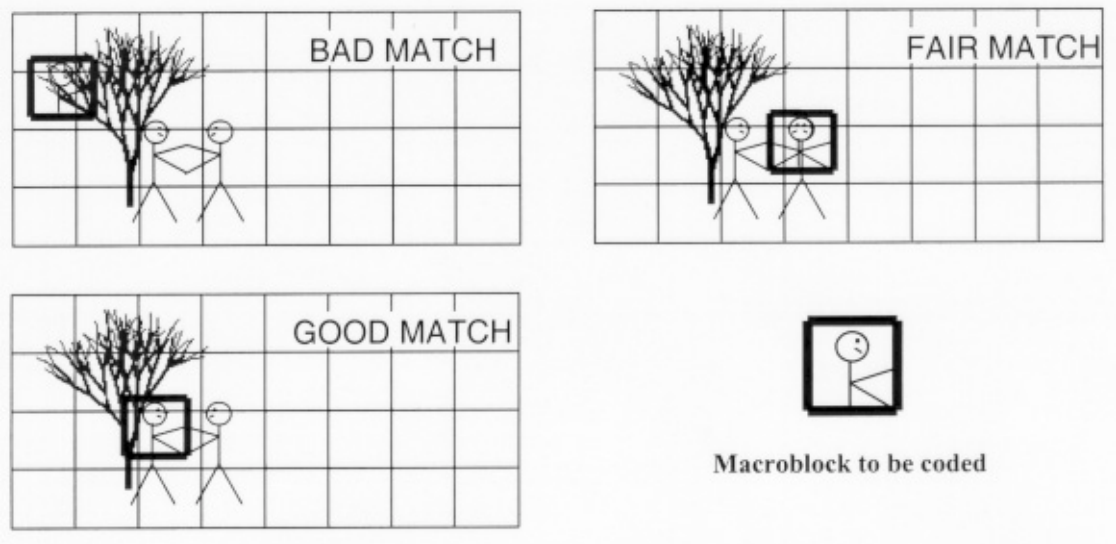


Figura 16

Achada a melhor escolha, o codificador utilizará vetores de movimento (*motion vectors*) ao macrobloco, indicando quão horizontalmente e verticalmente o macrobloco precisará ser movido.

O processador de redundância temporal utilizará o macrobloco pertencente ao quadro 1 para gerar uma representação no quadro 2 que contenha apenas as diferenças existentes entre os dois quadros. Esse quadro diferença é chamado de erro residual.

Se os dois quadros possuírem alto grau de redundância temporal, então o quadro diferença produzido pelo processador possuirá um número elevado de pixels cujos valores serão próximos de zero. Por outro lado, se o quadro 2 for completamente diferente do quadro 1, o codificador não encontrará macroblocos correspondentes e a estimação de movimento não será utilizada, ou seja, este quadro será codificado como um intra-quadro.

## Decodificação

Decodificação é o processo inverso da codificação. Para decodificar uma seqüência de bits gerada pelo codificador MPEG, é necessário reverter a ordem do processo de codificação.

Um decodificador MPEG consiste de uma entrada de bits no buffer, um decodificador de comprimento variável *(variable length decoder VLD*), um quantizador inverso, uma transformada inversa discreta do coseno (IDCT), uma interface de saída para o ambiente requerido (*hard drive*, buffer de vídeo, etc.) e suporte para compensação de movimento. Esse decodificador é mostrado na figura abaixo.

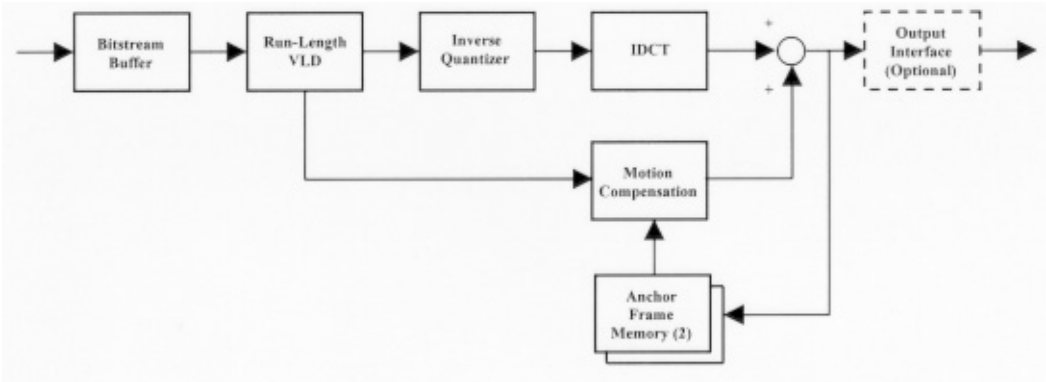


Figura 17

A entrada da seqüência de bits de um buffer consiste de uma memória que opera de maneira inversa ao buffer no codificador. Para aplicações onde a taxa de bits é fixa, o fluxo constante de bits é levado à memória e lido a uma taxa variável.

O decodificador de comprimento variável *(variable length decoder VLD)* é provavelmente o equipamento mais caro do decodificador porque ele opera no chamado *bit-wise basis.* Ele precisa "olhar" para cada bit na seqüência de bits codificada, a fim de determinar o fim de um bloco e o início de outro.

O bloco quantizador inverso (*inverse quantizer block*) fornece os coeficientes DCT sem quantização que serão utilizados pelo bloco IDCT.

O bloco IDCT vai aplicar a transformada inversa discreta do coseno nesses coeficientes, recuperando assim a informação original, no domínio espacial. Temos, então, a imagem original recuperada.

## Conclusões

Os padrões MPEG e seus algoritmos foram desenvolvidos por uma equipe internacional de especialistas. Seus arquivos são pequenos e, de uma forma geral, de boa qualidade. Por ser um padrão internacional e aberto, é usado na maioria das máquinas. Além disto, existe uma grande quantidade de arquivos MPEG disponíveis na Internet. Vários equipamentos de hardware, como placas gráficas e de vídeo de diversos fabricantes e diversos sistemas operacionais como o Windows 95, Windows NT e Macintosh OS suportam o MPEG. O padrão MPEG-1, por exemplo, funciona em 90% das máquinas. O MPEG-2, padrão escolhido para o DVD *(digital versatile disk)*, apesar de só rodar atualmente em 5% das máquinas, está com uma expectativa de mudança desta proporção para os próximos 2 anos, quando o DVD tornar-se amplamente utilizado.

Os padrões MPEG também possuem limitações. Em algumas situações onde o equipamento não é suficientemente rápido, é percebido problema no sincronismo entre o áudio e o vídeo. Além disto, a qualidade do vídeo depende do sistema em questão, ou seja, quanto mais antiga for a máquina, pior ficará a apresentação do vídeo MPEG. Estas falhas aparecem devido aos seguintes motivos: A codificação dos quadros P e B requerem alto poder computacional e a matriz de quantização precisa ser armazenada e transmitida junto aos quadros comprimidos, provocando queda na taxa de compressão.

Finalmente, quando utilizamos as técnicas de codificação com perdas, podemos perceber que a perda de informações relativas a alta freqüência provocam um borrado nas bordas dos objetos presentes na imagem. Isto acontece devido a filtragem das altas freqüências no espaço da Transformada Discreta do Coseno.

Muitos vêem o MPEG como o formato mais popular no futuro, não apenas para arquivos de vídeo para computadores, mas também para o chamado *home entertainment*. Nesta visão futurista, o DVD substituirá o VHS e a TV Digital substituirá a TV analógica. Os formatos deverão caminhar juntos, como já ocorre com o Quicktime, que utiliza o padrão MPEG-4. E as grandes companhias como Sun, Microsoft, IBM e Apple desenvolverão juntas os futuros formatos MPEG.

## Bibliografia

[1]"Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s," *ISO/IEC 11172-2: Video* (November 1991).

[2]"Generic Coding of Moving Pictures and Associated Audio Information: Video," ISO/IEC 13818-2 : Draft International Standard (November 1994).

[3]Barry G. Haskell, Atul Puri, Arun N. Netravali, *Digital Video: An Introduction to MPEG-2*, Chapman and Hall, 1997.

[4]K.R. Rao, P. Yip, *Discrete Cosine Transform – Algorithms, Advantages, Applications*, Academic Press, Inc., 1990.

[5]Majid Rabbani, Paul W. Jones, *Digital Image Compression Techniques*, SPIE Optical Engineering Press, 1991.

[6]Joan L. Mitchell, William B. Pennebaker, Chad E. Fogg, Didier J. LeGall, *MPEG Video Compression Standard*, Chapman and Hall, 1997.

[7]*IEEE Micro Magazine – Media Processing*, IEEE Computer Society, Volume 16 Number 4, August 1996.

## Referências

http://members.aol.com/symbandgrl/
http://www.mmig.ufp.pt/~nribeiro/sebentas/ecom/doc/crm06.pdf
http://pessoa.ufp.pt/~nribeiro/sebentas/ecom/doc/aulas.html
http://bmrc.berkeley.edu/research/mpeg/mepg-overview.html
http://www.org/mpeg/video.html#video-overview
http://www2.netpe.com.br/users/rlima/mp3.htm
http://www.megavizyon.com/mpeg.htm
http://www.whatis.com/mpeg.htm
http://www.drogo.cselt.it/mpeg
http://drogo.cselt.stet.it/mpeg/#about_mpeg.htm
http://alf.fe.up.pt/aefewww/old/deec208/relatorio.html
http://www.rahul.net/jfm/image.html
http://152.84.253.20/color_faq.html#definitions
http://www.realmagic.com/faq_streaming_mpeg_video.htm
http://www.lorraine-internet.com/jag/dossier/mpeg/index.html
http://www.gta.ufrj.br/~vidal/mpeg/mpeg.html
http://www.inf.puc-rio.br/~refletor/rt/indices.html
http://www.mariomarino.com.br/digital/formatos.htm
http://www.dcc.ufba.br/~thomas/assuntos/mpeg.html
http://www.citi.pt/multimidia/mpeg.html
http://www.msb.br/~jluiz/cursos/volp2/sld052.htm
http://www.medialab.je.up.pt/alunos/alopes/jpeg2.htm
http://www.cis.ohio-state.edu/~jain/cis788-99/compression/index.html
http://wwwam.hhi.de/mpeg-video/papers/sikora/mpeg1_2/mpeg1_2.htm