O PADRÃO JPEG E O SEU USO NA TRANSMISSÃO DE IMAGENS ADAPTATIVAS AO RETARDO NA REDE

Claudio G. Mello {cgmello@ime.eb.br}
Instituto Militar de Engenharia (IME)
Departamento de Engenharia de sistemas
Praça General Tibúrcio, 80

ABSTRACT

This work describes one of the most popular and powerful image compression techniques: JPEG. It's a lossy method, meaning that a descompressed image will be slightly different than the original, but keeps the most visually important part of an image while discarding information that are likely to be missed. It shows also, how to use its encoding properties to dinamically change the size of an image during transmission, over a communication channel, due to network conditions.

I – INTRODUÇÃO

O JPEG (Joint Photographic Experts Group) é um dos padrões mais populares e poderosos quando nos referimos à técnicas de compressão de imagens.

Sua história começa em 1982, quando a ISO (International Standards Organization) formou o Photographic Experts Group (PEG) com o objetivo de desenvover pesquisas na área de transmissão de vídeo, imagens e texto através de uma rede digital de serviços integrados (ISDN).

Em 1986, um subgrupo do CCITT (International Telegraph and Telephone Consultative Committee) já fazia pesquisas na área de compressão de imagens para transmissão de fax. E em 1987, os dois grupos se juntaram e formaram o JPEG (Joint PEG) para gerar padrões na área de compressão de imagens.

Até hoje o padrão JPEG é muito usado. Ele permite uma compressão de imagens "true-color" (24-bits) com grande rapidez e eficiência.

II – PROCESSO DE CODIFICAÇÃO

O método de compressão de uma imagem usado pelo JPEG não consiste de um algoritmo, mas de um conjunto de operações que são executadas em série sobre a imagem original. É possível regular a taxa de compressão de tal forma a conseguir uma imagem de tamanho o menor possível, mas que, por outro lado, terá uma perda muito alta na qualidade final. Ou optar por uma imagem de alta fidelidade e ainda assim obter um tamanho final menor que o

tamanho da imagem original.

A técnica usada pelo JPEG difere de outros padrões de compressão de imagens (como o RLE, LZW, etc.) principalmente pelo fato de ser um processo com perda das informações originais, ou seja, uma imagem JPEG descomprimida será ligeiramente diferente da original. Os outros métodos de compressão sem perda garantem a volta à imagem original, pois não descartam dados no processo de compressão da imagem.

Este é um dos motivos pelos quais o JPEG consegue alcançar índices maiores de compressão. Ele elimina informações consideradas menos importantes. Os dados descartados são aqueles que não são facilmente distinguíveis pelo olho humano, ou seja, têm pouca importância na definição da imagem.

As informações desprezadas pelo JPEG são principalmente aquelas pequenas diferenças de tonalidade de uma cor em certas áreas da imagem, pois são pouco importantes para a composição da imagem do ponto de vista de nossa visão. Por isso, o padrão JPEG tem bons resultados com imagens derivadas de fotografias, vídeos e paisagens. Mas não conseguirão comprimir bem documentos em preto-e-branco, desenhos a caneta e figuras compostas de linhas bem definidas de modo geral.

O JPEG é um excelente padrão para se armazenar imagens 24-bits, como nas aplicações multimídia. Cada pixel é definido por 3 bytes, um byte para cada intensidade do espectro RGB (red, green, blue) resultando em uma imagem de alta definição com até 16 milhões de cores (Figura 1).

Figura 1 - Pixel "true-color" de 24-bits

A taxa de compressão conseguida depende do tipo da imagem. Uma figura com qualidade fotográfica, por exemplo, pode ser comprimida a uma razão de 20:1, ou seja, apenas 5% do tamanho da imagem original, sem perda significativa da qualidade visual da imagem. Taxas mais altas de compressão irão degradar cada vez mais a qualidade da imagem, mantendo, entretanto, sempre a maior identidade visual possível com a imagem original.

Esta taxa de compressão não apenas economiza espaço em disco para o armazenamento, mas também beneficia diretamente os tempos necessários para a transmissão destas imagens, que era o objetivo principal do grupo JPEG. Por isso, é um dos mais populares padrões de codificação de imagens na Internet, e nas redes em geral, principalmente no caso de figuras maiores.

O processo de codificação JPEG permite que você regule facilmente a razão compressão x qualidade da imagem final. Ou seja, quanto maior a compressão, pior a qualidade final, e vice-versa. É uma escolha de custo-benefício. Este fator é normalmente chamado de "fator de qualidade" (Q). O fator Q varia de 1 a 100. Para Q=1 temos a maior compressão possível com a pior qualidade de imagem. E Q=100 gera uma imagem o mais próxima da original possível, mas de maior tamanho. O ideal é encontrar o menor valor de Q para o qual a imagem seja perfeitamente aceitável, o mais idêntica à imagem original. O fator mais comumente utilizado é Q=75.

Mas o JPEG não é sempre a solução ideal. Alguns motivos:

- Certas imagens não são tão comprimíveis, como as já citadas anteriormente (desenhos, documentos, etc.);

- O processo de codificação é extremamente lento se implementado todo em software. Para soluções que exijam velocidade, uma implementação baseada em hardware dedicado é necessária;

- O padrão JPEG não é fácil de ser implementado;

- Os arquivos de imagens JPEG (.jpg) ainda não são lidos por todos os softwares do mercado (processadores de texto, por exemplo).

O processo de codificação do JPEG é baseado na transformada discreta do cosseno (DCT). Os algoritmos baseados na DCT são geradores de perdas de informação por natureza e serão detalhados mais adiante.

De modo global, podemos esquematizar o processo de codificação dividido nos seguintes estágios:

i. Transformar o espaço de cores da imagem;

ii. Amostrar as componentes de crominânica;

iii. Aplicar a DCT a blocos de 8x8 pixels;

iv. Usar uma matriz de quantização otimizada para o olho humano;

v. Comprimir os dados

Estes estágios estão resumidos na Figura 2 [1].

i. Transformar o espaço de cores

Uma imagem em tons de cinza, onde a intensidade de cada pixel varia de 0 a 255 (representação em 8 bits) com 0=preto e 255=branco, não possui espaço de cores, logo não precisa de transformação. Mas para imagens coloridas, o JPEG codifica cada componente do espaço de cores independentemente.

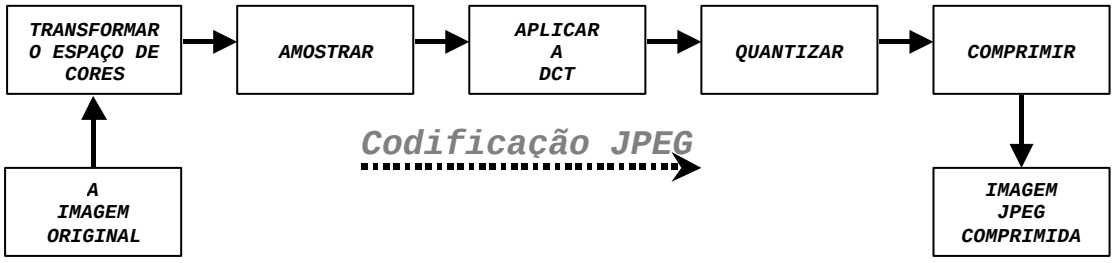

Figura 2 – O processo de codificação JPEG

O modelo de cores RGB (assim como o HIS, CMY, etc.) poderia ser codificado simplesmente como três componentes independentes, mas isto tornaria muito difícil o descarte das informações menos essenciais. Existe um modo mais eficiente de se fazer esta codificação, basta usar modelos de cores como o YUC, YCbCr, entre outros.

O modelo YCbCr, por exemplo, é utilizado pelos canais de TV na Europa. A componente Y é chamada de "luminância" (luminance), e não possui nenhum efeito sobre a cor do pixel, mas apenas sobre sua luminosidade, regula o brilho da imagem. As componentes Cb e Cr são denominadas "crominância" (chrominance), e juntas definem a cor de cada pixel. Informalmente falando, as componentes Cb e Cr representam, respectivamente, a relação entre o azul e o vermelho em relação ao verde.

A maior parte das informações mais sensíveis ao olho humano encontra-se na componente Y. As outras componentes Cb e Cr são menos importantes para a formação da imagem, do ponto de vista de nossa visão, logo aquela componente terá prioridade no descarte de informações para efeito de compressão.

Então, neste estágio precisamos transformar o espaço de cores para um modelo do tipo luminância/crominância para termos uma maior flexibilidade na separação entre as informações mais úteis e as menos úteis do ponto de vista da percepção da visão humana. Podemos obter a cor YCbCr de um pixel RGB pelas seguintes equações:

$$\boldsymbol{Y} = 0.2990*\boldsymbol{R} + 0.5870*\boldsymbol{G} + 0.1140*\boldsymbol{B};$$

$$\boldsymbol{Cb} = -0.1684*\boldsymbol{R} - 0.3316*\boldsymbol{G} + 0.5000*\boldsymbol{B};$$

$$\boldsymbol{Cr} = 0.5000*\boldsymbol{R} - 0.4187*\boldsymbol{G} - 0.0813*\boldsymbol{B};$$

Onde Y irá assumir valores entre 0 e 255, e as componentes Cb e Cr valores entre -128 e +128. Inclusive, a componente de luminância Y por si só nos dá um método de transformar um imagem colorida em tons de cinza (grayscale). Na Figura 3 temos um exemplo de como as componentes do modelo YCbCr agem sobre uma imagem.

Figura 3-(a)

Figura 3-(b)

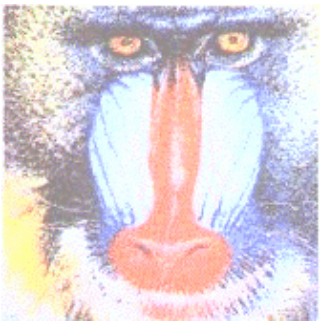

Figura 3-(c)

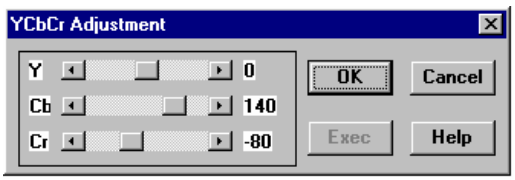

Figura 3-(d)

Figura 3 – Exemplo de manipulação de uma imagem no modelo YCbCr. (a) Imagem original de um babuíno; (b) Variação em CbCr (cor); (c) Variação em Y (brilho); (d) Tela de ajuste do modelo YCbCr (LviewPro ã [9]).

ii. Amostrar a crominância

Devemos aproveitar o fato da componente de crominância não ser tão importante na definição da imagem para o olho humano. Então, vamos eliminar, artificialmente, informações apenas desta dada componente por meio de uma amostragem pela média dos valores da crominância para cada pixel tomados 2 a 2.

Se, por exemplo, tivermos uma imagem de 1000x1000 pixels, vamos considerar a totalidade dos valores para a luminância, ou seja, teremos uma matriz de 1000x1000 tons de cinza para esta componente.

E para cada componente Cb e Cr da crominância, vamos amostrar a uma taxa de 2:1 na horizontal, ou seja, para cada dois valores na horizontal da matriz da crominância vamos armazenar apenas um ponto cujo valor será a média destes dois valores.

Na vertical, temos dois tipos de implementação. Podemos varrer normalmente cada linha apenas amostrando na horizontal (padrão 2h1v) ou podemos também levar em consideração os valores na vertical para o cálculo da média, aumentando ainda mais o descarte de informações (padrão 2h2v).

O padrão 2h1v é mais conhecido devido a sua implementação para o padrão de TV chamado de NTSC (National Television Standards Committee). Utilizando este padrão conseguimos de imediato uma diminuição de 50% nas matrizes das componentes Cb e CR, sem alteração significativa na qualidade da imagem. Ou seja, uma diminuição de 33,33% no tamanho da imagem se levarmos em conta que a matriz da componente de luminância permaneceu intacta.

iii. Aplicar a DCT

Neste passo, cada matriz do modelo YCbCr da imagem é divida independentemente em blocos de 8x8 valores (pixels). Então a DCT é aplicada a cada bloco da imagem, convertendo a representação da imagem em uma mapa 8x8 de frequências. Os termos mais à esquerda e mais acima representam as frequências baixas (ondas grandes e lentas), são os valores médios do bloco. Enquanto que à medida que vamos nos aproximando dos termos mais à direita e mais abaixo, temos os elementos que representam as frequências altas (ondas curtas e rápidas), ou seja, as transições rápidas de tons que ocorrem no bloco em pixels adjacentes.

A DCT unidimensional de uma sequência de pontos $\{u(n);\ 0 \le n \le N-1\}$ é dada por [3]:

$$c(k,n) = a(k).\cos\frac{p(2n+1)k}{2N}, \quad com$$

$$v(k) = \sum_{n=0}^{N-1} u(n).c(k,n), \quad (0 \le k \le N-1)$$

onde

$$a(k) = \sqrt{\frac{2}{N}}, \quad para \;\; 1 \le k \le N-1$$

$$a(0) = \sqrt{\frac{1}{N}} \quad e$$

Então, a DCT bi-dimensional [2] sobre um bloco *N*x*N* de uma imagem cuja componente Y possua os valores *i*(*x*,*y*), com *x*,*y* ∈ [0,*N*-1], será definida por uma DCT unidimensional sobre as linhas do bloco, e em seguida por outra DCT unidimensional sobre as colunas de valores da primeira DCT. Teremos a seguinte equação para os *N*x*N* coeficientes *I*(*u*,*v*) da DCT:

Este é o passo mais custoso do processo de codificação de uma imagem JPEG. Na verdade, teríamos que

$$I(u,v) = \sum_{x=0}^{N-1} (\sum_{y=0}^{N-1} i(x,y).c(x,u).c(y,v)),$$

$$para \; u,v \in [0, N-1]$$

executar um grande número de operações devido à multiplicação das matrizes:

```
for (u=0; u<8; u++)
    for (v=0; v<8; v++)
    {
        for (x=0; x<8; x++)
            for (y=0; y<8; y++)
                DCT[u][v] += imagem[x][y] / 8 * cos ((PI * u * (2*x+1))/16) * cos ((PI * v *
                (2*y+1))/16);
        if (u>0) DCT[u][v] *= sqrt(2);
        if (v>0) DCT[u][v] *= sqrt(2);
    }
```

Mas graças a grande simetria da matriz da transformada, o número de operações se reduz drasticamente, tornando a DCT viável. Na verdade, teríamos 1024 operações na DCT bi-dimensional, ou seja, no cálculo da DCT unidimensional, para cada elemento *v*(*k*) são executadas 8 operações, sendo que cada linha possui 8 elementos e que essa operação é repetida nas 8 linhas, chegamos ao número (8x8x8). Como temos que repetir a DCT agora nas colunas para obter a bi-dimensional, então multiplicamos por 2 (8x8x8x2=1024). Devido a simetria, tem-se que o número de operações para a DCT unidimensional é *O(N.logN)*, e na prática temos algo em torno de 80 operações para a DCT bi-dimensional.

Na Figura 4(a) temos um exemplo de um bloco de 8x8 pixels da componente Y (0-255) de uma imagem.. Aplicando a DCT a este bloco teremos a matriz da Figura 4(b).

182	194	201	169	168	187	175	169
176	146	161	182	157	141	156	174
150	147	141	150	150	140	135	146
106	106	126	116	116	129	129	140
93	94	78	87	85	112	116	94
73	121	106	95	88	71	87	77
22	22	29	22	21	20	20	26
2	3	3	2	3	3	3	3

Figura 4-(a)

818	2	3	-14	-7	2	0	-1
460	5	4	-1	10	7	11	0
-69	24	-1	-4	4	0	8	4
73	17	0	-4	-30	-4	-9	0
-19	-22	7	17	-23	1	11	-1
-6	-1	-3	-6	-14	-12	13	6
48	23	-9	-3	-18	-18	-11	0
-31	-1	6	25	-3	0	0	-5

Figura 4-(b)

Figura 4 – (a) Bloco 8x8 da imagem original (componente Y). É um pedaço aparentemente mais claro na parte baixa e mais escuro em cima; (b) Matriz DCT resultante.

Como ilustração, vamos calcular o valor do coeficiente da DCT *I(0,0)* para a matriz da Figura 4(a):

$$c(x,0) = a(0).\cos\frac{p(2x+1).0}{2.8} = \frac{1}{\sqrt{8}}$$

$$c(y,0) = a(0).\cos\frac{p(2y+1).0}{2.8} = \frac{1}{\sqrt{8}}$$

Logo,

$$I(0,0) = \sum_{x=0}^{7} (\sum_{y=0}^{7} i(x,y).\frac{1}{\sqrt{8}}.\frac{1}{\sqrt{8}})$$

$$= \frac{1}{8}.\sum_{x=0}^{7} \sum_{y=0}^{7} i(x,y) = \frac{1}{8}.6546 = 818$$

O que temos ao final da aplicação da transformada é um mapa de frequências. São 64 coeficientes que descrevem a mesma imagem.

Para reconstruir o mapa de pontos da matriz original, basta somar as ondas multiplicadas pelos coeficientes DCT, que é a transformada inversa (Figura 5):

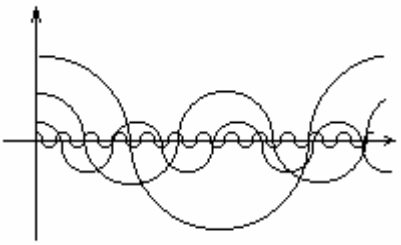

Figura 5 - Reconstrução a partir dos coeficientes da DCT

Então, devemos descartar [7] as frequências mais altas, que representam transições rápidas entre pixels adjacentes, ou seja, pequenos detalhes ou até mesmo ruídos na imagem e que devem ser eliminados, pois o olho humano quase não detecta estas ondas de oscilações rápidas, e preservar as baixas frequências, que são importantes para a definição da imagem.

iv. A matriz de quantização

Com a finalidade de descartar uma certa quantidade de informação, o processo de codificação JPEG inclui este passo chamado de quantização [4]. Agora, cada valor da matriz 8x8 resultante da aplicação da DCT sobre cada um dos componentes YCbCr será dividido por um elemento q_{XY} de uma matriz de quantização também 8x8, e o resultado é truncado, tomando-se apenas a parte inteira da divisão.

Logo, quanto maior for o valor do quoficiente q_{XY}, maior a probabilidade da divisão inteira ter resultado 0, e maior a perda de informação.

Por exemplo, seja D_{XY}=22 um elemento da matriz DCT a ser quantizada. E seja q_{XY}=7 o elemento equivalente na matriz de quantização. Então, o resultado da divisão INT(22/7) será igual a 3.

Para retornarmos ao valor original, basta multiplicar este valor pelo q_{XY}=7, ou seja, teremos 3x7=21, um erro $E \leq q_{XY}$-1. Se aumentarmos o coeficiente de quantização para q_{XY}=29, por exemplo, o resultado da divisão INT(22/29) será igual a 0, e a imprecisão no momento de se achar o valor original será muito maior.

Com isso temos o poder de regular o número de valores iguais resultantes das divisões inteiras, pois:

INT(29/29)=INT(30/29)=…=INT(57/29) = 1

assim como a quantidade de zeros que será gerada (grau da matriz esparsa).

Com esse método simples e eficiente temos uma maneira de regular a compressão dos dados, descartando criteriosamente informações.

Cada uma das 64 posições da matriz de quantização possui um coeficiente q_{XY} diferente. De acordo com a DCT, devemos ter coeficientes mais baixos para os termos mais à esquerda e mais acima, de modo a preservá-los. E coeficientes mais altos para os termos mais à direita e mais abaixo, de modo a conseguirmos uma perda maior de informação nestes valores da DCT.

Também deve existir um tratamento diferenciado entre as componentes YCbCr. Os coeficientes da matriz de quantização a ser usada na matiriz da DCT resultante sobre a componente de luminância (Y) deve ser proporcionalmente maior que os mesmos das matrizes de quantização que serão usadas sobre as componentes de crominância (Cb,Cr). Isto permite selecionar a perda em direção a componente menos importante (crominância).

Temos o que chamamos de fator Q de qualidade. Este fator Q é um número inteiro que varia entre 0-100. O JPEG inicializa a matriz de quantização com coeficientes que resultem em uma imagem de boa qualidade, que é o equivalente ao fator Q=75. Então, vai aumentando ou diminuindo os coeficientes da matriz simultâneamente e proporcionalmente ao fator Q.

Quanto mais alto o valor de Q, menores serão os coeficientes q_{XY} resultando em uma maior qualidade final da imagem, e maior tamanho da imagem.

E para valores menores de Q, maiores serão os coeficientes, piorando a qualidade da imagem, mas em compensação, atingindo altas taxas de compressão. Os valores práticos de Q situam-se entre $5 \leq Q \leq 95$, sendo que o valor padrão para um resultado de boa qualidade e compressão é Q=75.

A Figura 6 mostra alguns exemplos de compressão com fatores Q=5 e Q=75.

As matrizes de quantização correspondentes ao fator Q escolhido são armazenadas junto com os dados da imagem para posterior reconstrução da imagem original.



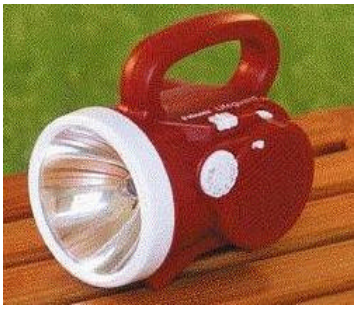

Figura 6-(a) **Figura 6-(b)** **Figura 6-(c)**

Figura 6 – (a) Imagem original de uma lanterna de tamanho 250x215 pixels “true-color” (180KB); (b) Imagem JPEG com Q=5 (3KB); (c) Imagem JPEG com Q=75 (18KB)

Na Figura 7(a) temos um exemplo de uma matriz de quantização. E na Figura 7(b) a matriz resultante desta quantização sobre a matriz DCT da Figura 4(b) pela equação:

DCT[u][v] = (int)(DCT[u][v]/quant[u][v]);

3	5	7	9	11	13	15	17
5	7	9	11	13	15	17	19
7	9	11	13	15	17	19	21
9	11	13	15	17	19	21	23
11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29
17	19	21	23	25	27	29	31

Figura 7-(a)

272	0	0	-1	0	0	0	0
92	0	0	0	0	0	0	0
-9	2	0	0	0	0	0	0
8	1	0	0	-1	0	0	0

-1	-1	0	1	-1	0	0	0
0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0
-1	0	0	1	0	0	0	0

Figura 7-(b)

Figura 7 – (a) Matriz de quantização; (b) Matriz quantizada a partir da matriz DCT da Figura 4(a).

É desta maneira que adaptamos o tamanho da imagem de acordo com as nossas necessidades.

v. Compressão dos dados

O resultado final da quantização gera vários valores redundantes, inclusive muitos zeros. Agora, devemos armazenar estes dados usando algum algoritmo de compressão [8].

O mais usado é o algoritmo de Huffman. Este algoritmo é sem perdas e na codificação JPEG ele é implementado de uma maneira especial. Os dados são lidos da matriz em zig-zag conforme a Figura 8 (de cima para baixo, vindo da esquerda para a direita), e quando chegam num ponto da matriz onde só teremos zeros dali para diante, a sequência termina com uma marca de *END.*

Os dados seriam alinhados da seguinte maneira:

272, 0, 92, -9, 0, 0, -1, 0, 2, 8, … END

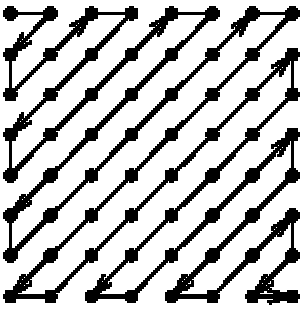

Figura 8 – Os elementos da matriz são lidos em zig-zag.

Cada elemento então é associado a uma frequência. Por exemplo, o número 0 aparece 48 vezes, o -1 aparece 6 vezes, o 1 quatro vezes, e o 2, 3, 8, -9, 92 e o 272 aparecem apenas uma vez. Então é montada a árvore por ordem de frequências. Cada ramo da esquerda é rotulado com 0, e os da direita com 1. Os elementos que aparecem o menor número vezes ficam mais abaixo na árvore conforme a Figura 9:

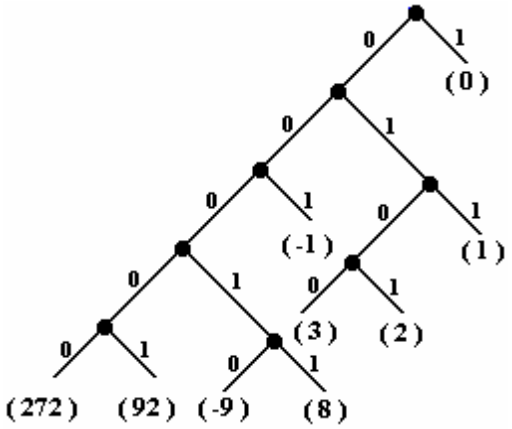

Figura 9 – Árvore de frequências

Logo, atribuímos a cada número um código na tabela de Huffman (Figura 10).

Então, em zig-zag, os dados armazenados ficaram da forma:

0000010000100010110011010100011 1…END

Se fossemos armazenar os números sem nenhum método de compressão o tamanho do bloco original seria de 64 x 8 bits = 512 bits, pois cada um dos 64 números seria representado por 8 bits (0-255). Com Huffman armazenamos este mesmo bloco com apenas 91 bits mais o código END e a tabela de códigos.

Número	Código de Huffman
0	1
-1	001
1	011
2	0101
3	0100
8	00011
-9	00010
92	00001
272	00000

Figura 10 - Código de Huffman

Estima-se que o tamanho da imagem JPEG fique em torno de 5% a 10% do tamanho da imagem original "true-color" (66,67% pelo processo de amostragem, e depois um decréscimo em cima desse valor de 10% a 12% ao final da compressão), o que é um ótimo ganho para o armazenamento do JPEG, que será então encapsulado em seu formato de arquivo próprio, ou para efeito de diminuição no tempo de transmissão.

III - LEITURA DE UMA IMAGEM JPEG

A leitura da imagem codificada pelo JPEG consiste nas operações inversas do processo de codificação (Figura 2), e consiste dos seguintes passos:

- Descompressão dos dados usando Huffman, a partir da tabela de códigos recuperada;
- Desquantização dos dados a partir da matriz de quantização armazenada junto da imagem;
- Aplicação da DCT inversa:
- Desamostragem dos dados da crominância;
- Transformação do espaço de cores YCbCr em outro qualquer tipo RGB.

IV - TRANSMISSÃO DE IMAGENS

Com o aumento da capacidade das redes atuais, e da velocidade dos computadores, a transmissão em tempo real de imagens, som e vídeo está se tornando cada vez mais passível de ser implementada em aplicações do tipo vídeofone, vídeoconferência, whiteboard, vídeo por demanda, e outras.

Mesmo com as redes de alta velocidade mais recentes, é essencial o uso da compressão das informações a serem transmitidas por um canal de comunicação, assim como o controle de fluxo para evitar congestionamentos. Para transmissão de vídeo, alguns dos padrões usados são listados a seguir, além de existirem outros padrões proprietários desenvolvidos especificamente para algumas aplicações como a NV (Network Video), uma aplicação de videoconferência criada pela Xerox, baseada na tranformada de Haar, que tem um custo computacional menor que a DCT, apesar de alcançar taxas mais baixas de compressão.

i. H.261

O padrão H.261 descreve um complexo algoritmo de compressão de vídeo que alcança altas taxas de compressão. Ele foi projetado para uso em ISDN (Integrated Services Digital Network), isto é, para redes com canais de comunicação fixos em (p x 64 kb/s, $p \in [1,32]$.

O processo de codificação de blocos de 8x8 pixels consiste em :

- Detecção de movimento.

- Aplicação da DCT;
- Quantização;
- Compressão por Huffman.

Depois de calcular o movimento relativo entre o quadro atual e o anterior, o algoritmo decide entre codificar apenas a diferença entre os quadros, ou simplesmente codificar um novo quadro. É o que se chama de codificação INTER-FRAME e INTRA-FRAME, respectivamente.

Então, os dados são transformados em coeficientes pela DCT, quantizados e comprimidos por Huffman.

ii. MPEG

O MPEG (Moving Picture Experts Group) define uma padrão de codificação de vídeo e o correspondente áudio. O MPEG alcança taxas maiores de compressão que o JPEG devido ao uso de quadros intermediários que registram apenas as mudanças entre duas imagens da sequência de vídeo.

Existem 3 tipos de quadros [6]:

- *Intracoded* (I) – simplesmente codificado, sem a influência de outros quadros;
- *Predicted* (P) – quadros que representam apenas as mudanças entre o quadro anterior e o atual;
- *Bidirectional* (B) – este quadro é codificado levando-se em conta o quadro anterior e o posterior ao atual.

Uma sequência típica de quadros MPEG é: (IBBPBBPBB)(IBBPBB… Ou seja, é enviado um quadro completo e depois apenas as mudanças entre os quadros subsequentes. E após a transmissão de um certo número de quadros intermediários, um novo quadro completo é enviado de forma a evitar a propagação de erros, e então, o processo se repete.

Também são usadas a DCT, a quantização e a compressão de Huffman no processo de codificação.

iii. MJPEG

O padrão MJPEG (Motion JPEG) trata cada quadro de uma sequência de vídeo como uma imagem JPEG, e executa a completa codificação independentemente dos quadros anterior e posterior. Ou seja, é a transmissão contínua de imagens JPEG.

É usado pelos sinal de TV NTSC e PAL-M.

A desvantagem é em termos de taxa de compressão, menor em relação ao MPEG devida ao fato do MJPEG não usar quadros intermediários com às diferenças entre as imagens.

A vantagem é que você tem todos os quadros integrais da sequência de vídeo, o que permite a edição. E o fato de que a codificação MPEG tem um custo computacional elevado devido principalmente à predição das diferenças entre quadros.

O MJPEG também é indicado para aplicações em que os quadros da sequência de vídeo tenham pouca

correlação entre si, como whiteboard e monitoramento interno/externo por TV, onde câmeras tiram fotos de lugares alternados periodicamente.

Ao se utilizar redes de pacotes como a Internet, não temos garantia de qualidade de serviço (QoS), por exemplo, a garantia de um canal de 64 kbps. Mesmo assim, podemos criar métodos de controle de congestionamento para prevenir um colapso do recurso.

A idéia consiste em se criar mecanismos de controle de fluxo no transmissor agindo sobre o tamanho das imagens que são transmitidas de acordo com as condições da rede, principalmente a métrica do retardo ida e volta de um pacote na rede, ou seja, o tempo que um pacote leva para chegar ao seu destino e voltar confirmando o recebimento pelo receptor. Na medida do retardo também influe, implicitamente, a taxa de perda de pacotes, o que causa retransmissões.

Quando o transmissor enviar o pacote 1, ele espera receber a confirmação (ACK) daquele pacote com a requisição do próximo (ACK 2), conforme a Figura 11 [5].

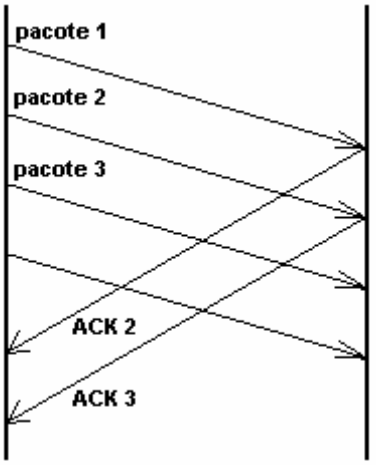

Figura 11 – Esquema de fluxo de pacotes

Normalmente nos protocolos com controle de erro, quando um pacote *n* se perde na transmissão, a sua confirmação com requisição (ACK *n*+1) não é recebida. Isto indica que o pacote deve ser enviado novamente. No caso de transmissão de vídeo isto não étão levado em conta, pois é melhor perder informação em um quadro do que interromper a sequência de vídeo esperando o pacote que se perdeu ser retransmitido. Então, neste caso as confirmações são usadas apenas para o cálculo do retardo e para a análise das condições da rede.

Vamos supor uma aplicação de transmissão de imagens em tempo real usando MJPEG. A imagem tem dimensões de 100x100 pixels, e uma taxa de visualização de 10 fps (10 quadros por segundo). O que é uma taxa bem pequena, visto que a televisão usa 30 quadros/s.

Se codificarmos a imagem “true-color” como uma mapa de bits, sem nenhum tratamento de compressão, o tamanho da imagem seria de 100x100x24 bits ~ 30kb.

Codificando com JPEG e usando o fator de qualidade Q=75, o tamanho da imagem se reduz a algo em torno de 2kb (de 5 a 10% do tamanho original).

Para uma taxa de 10 fps, temos 20 kbps. Ou seja, uma conexão via MODEM de 28800 bps seria suficiente.

Então, iniciamos a transmissão neste canal de 28.8 Kbps e periodicamente fazemos uma análise do retardo médio dos pacotes. Caso haja um congestionamento na rede, ou seja, os pacotes estejam demorando muito para chegar no destino, devemos adotar uma estratégia de controle de fluxo.

A estratégia inicial pode ser primeiro ir decrementando a taxa de visualização da sequência de imagens, avisando ao receptor que a taxa será de 9, 8, 7, 6 ou 5 fps, decrescendo conforme o aumento do retardo. Chegará um ponto em que não poderemos mais diminuir esta taxa. Então, mudamos de estratégia.

A solução passa a ser adaptar o tamanho da imagem a esta condição de congestionamento na rede, ou seja, vamos piorar a qualidade da imagem a ser transmitida, em favor da diminuição do tamanho da imagem (eventualmente, poderíamos até transmitir apenas a componente Y, ou seja, enviar a imagem apenas em tons de cinza).

Basta regular o fator Q do JPEG. Como iniciamos com Q=75, devemos variar este valor para baixo conforme o congestionamento se agrave.

É óbvio que para uma taxa de 5 fps e imagens codificadas com Q=10, por exemplo, o resultado da sequência das imagens será a pior possível. Mas isto pode servir como uma adaptação apenas temporária a um problema no canal, permitindo que não haja uma parada no serviço, ou mesmo uma desconexão por falta de QoS.

No momento em que o congestionamento for melhorando devido à ação do controle de fluxo, a aplicação começa a aumentar gradativamente a qualidade da imagem (Q), e depois a taxa de visualização (quadros por segundo).

Este foi um caso de fluxo altamente limitado. Para aplicações em redes de alta velocidade, os parâmetros seriam proporcionalmente maiores.

VI. CONCLUSÃO

Este trabalho procurou mostrar alguns dos métodos de codificação mais usados pelos padrões de imagem e vídeo como o JPEG, MPEG, H.261 e outros.

Mostramos os conceitos de transformação de espaço de cores, modelo de luminância/crominância, amostragem, transformada discreta do cosseno (DCT), quantização e compressão de Huffman. E como usar o fator Q para adaptar a transmissão de imagens de acordo com as condições de tráfego na rede.

Devemos ter em mente as imagens a que o padrão JPEG se destina, que são as imagens naturais ou do "mundo-real". Para outras imagens como as desenhadas com linhas bem definidas ou documentos em preto-e-branco, outros padrões devem ser estudados, como o GIF. Exploramos características da visão humana no descarte de informação, pois a codificação JPEG é com perda, mas uma perda seletiva e regulada pelo fator Q.

Com certeza o padrão JPEG ainda será muito popular e as novas aplicações continuarão a usar de seus recursos e conceitos envolvidos em seu processo de codificação.

REFERÊNCIAS

[1] James D. Murray and Willian vanRyper 1994, “Graphics File Formats” – O’Reilly & Associates

[2] Niblack, Wayne 1986, “An Introduction to digital image processing” – Prentice/Hall

[3] Anil K. Jain 1988, “Fundamentals of Digital Image Processing” - Prentice/Hall

[4] Rick Booth 1997, “Inner Loops: A Sourcebook for Fast 32-Bit Software Design” - Addison-Wesley Pub Co

[5] Thierry Turletti and Christian Huitema, “Videoconferencing on the Internet” – IEEE/ACM Transactions on Networking, vol.4, n.3, p. 340, June 1996

[6] Simons S. Lam, Fellow, IEEE, Simon Chow, and David K. Y. Yau, “A Lossless Smoothing Algorithm for Compressed Video” - IEEE/ACM Transactions on Networking, vol.4, n.5, p. 697, October 1996

[7] Watson, A. B., Solomon, J. A., & Ahumada, A. J., Jr. 1994, “The visibility of DCT basis” - IEEE Computer Society Press, 371-379

[8] Steven Amor 1995, “JPEG File Compression” - ITB235 Multimedia Systems

[9] Loureiro, L. 1993, “Lview for Windows” - Version 3.1, c 1993 Leonardo Haddad L.