Escola Politécnica da USP
Departamento de Engenharia de Sistemas Eletrônicos

Codificação de imagens segundo o padrão JPEG

Vítor H. Nascimento

Outubro de 2018

Nesta experiência serão vistos os princípios básicos do padrão JPEG de compressão de imagens incluindo transformação pela DCT e codificação de Huffman. O texto a seguir é baseado em [1].

1 Introdução

O padrão JPEG foi proposto inicialmente em 1992, e sua última versão data de 1994. É baseado principalmente em três pontos:

- No fato de que o número de receptores para cor no olho humano é menor do que o número de receptores para luminosidade, permitindo que a informação de cor seja armazenada com uma resolução menor do que a de luminância.
- Na observação de que os coeficientes da transformada de Fourier e extensões (principalmente a transformada discreta do cosseno) têm a maior parte dos valores mais elevados nas frequências baixas, o que permite que coeficientes correspondentes a frequências mais elevadas sejam quantizados com passos de quantização mais elevados;
- Em uma análise estatística da distribuição dos valores dos diversos coeficientes da transformada, permitindo a utilização de códigos eficientes para representá-los com um número baixo de bits.

Vamos ver cada um desses pontos com mais detalhes a seguir.

Antes de começar, no entanto, é importante ver uma maneira de medir a qualidade de um dado codificador. A medida mais importante na verdade é qualitativa: a menor distorção é aquela que uma pessoa média considera menor, e é difícil de definir critérios numéricos objetivos que aproximem bem um teste qualitativo. Vamos usar aqui um critério objetivo simples (que é comumente usado), o erro médio quadrático. Suponha que a imagem $h[n_1, n_2]$ tenha $N_1 \times N_2$ pixels, e que $\hat{h}[n_1, n_2]$ seja a imagem obtida após codificação e decodificação. Então

$$\text{MSE} = \frac{1}{N_1 N_2} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \left\| h[n_1, n_2] - \hat{h}[n_1, n_2] \right\|^2 . \tag{1}$$

É comum fornecer o MSE em dB, usando a *peak signal-to-noise ratio* (PSNR), que simplesmente compara o MSE com a máxima energia média possível em uma imagem (ou seja, supondo que todos os pixels são brancos):

$$\mathrm{PSNR} = 10 \log_{10} \left[\frac{(2^B - 1)^2}{\mathrm{MSE}} \right], \tag{2}$$

em que B é o número de bits usado para representar as intensidades (em geral, 8). Valores de PSNR maiores ou iguais a 30 dB em geral correspondem a uma qualidade de reconstrução razoável.

Para imagens coloridas em RGB, o MSE pode ser calculado somando-se os erros nas três cores, e dividindo o resultado por três.

2 Luminância e crominância

O primeiro ponto usado pelo padrão JPEG para reduzir o tamanho dos arquivos necessários para armazenar imagens é aproveitar algumas características do olho humano. Uma imagem $N_1 \times N_2$ no padrão RGB é representada por três matrizes de pixels $\boldsymbol{R}, \boldsymbol{G}, \boldsymbol{B}$, correspondentes às cores vermelho, verde e azul, cada uma com o mesmo número de pontos $N_1 \times N_2$ — são necessários portanto $3N_1N_2$ valores numéricos para representar a imagem.

O primeiro passo no padrão JPEG é transformar a representação para o padrão YCbCr, em que

$$\boldsymbol{Y} = \alpha_R \boldsymbol{R} + \alpha_G \boldsymbol{G} + \alpha_B \boldsymbol{B}, \tag{3a}$$

$$\boldsymbol{C}_b = \frac{1}{2(1-\alpha_B)} (\boldsymbol{B} - \boldsymbol{Y}), \tag{3b}$$

$$\boldsymbol{C}_r = \frac{1}{2(1-\alpha_R)} (\boldsymbol{R} - \boldsymbol{Y}), \tag{3c}$$

em que $\boldsymbol{Y}$ é o termo de *luminância*[1] (correspondente aos níveis de cinza), e $\boldsymbol{C}_b$ e $\boldsymbol{C}_r$ são os termos de *crominância*, e

$$\alpha_R = 0{,}299, \qquad \alpha_G = 0{,}587, \qquad \alpha_B = 0{,}114. \tag{4}$$

Os sinais de crominância, $\boldsymbol{C}_b$ e $\boldsymbol{C}_r$, são subamostrados por um fator de dois nas direções horizontal e vertical, pois o olho humano tem uma capacidade menor para distinguir detalhes de cores do que de luminância. Desta maneira, uma imagem colorida com $N_1 \times N_2$ pontos precisa de $1{,}5N_1N_2$ valores numéricos para ser representada. Considerando que o número de bits usado em média para cada canal seja o mesmo, tem-se uma redução pela metade na memória necessária.

3 Transformada discreta do cosseno (DCT)

O segundo passo é usar a propriedade de compactação de energia da DCT. Cada um dos sinais $\boldsymbol{Y}, \boldsymbol{C}_b, \boldsymbol{C}_r$ é transformado usando a transformada discreta do cosseno tipo 2 (DCT2),

[1] Veja [2] para alguns detalhes sobre a definição de luminância.

que é definida para cada um sinal genérico $h[n_1, n_2]$, definido no intervalo $0 \leq n_1 \leq N_1 - 1$, $0 \leq n_2 \leq N_2 - 1$ por

$$H[k_1, k_2] = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} h[n_1, n_2] a_{k_1}^{(N_1)}[n_1] a_{k_2}^{(N_2)}[n_2], \tag{5}$$

$$a_k^{(N)}[n] = \sqrt{\frac{s}{N}} \cos\left(\frac{(2n+1)k\pi}{2N}\right), \qquad s = \begin{cases} 1, & k = 0, \\ 2, & k \neq 0. \end{cases} \tag{6}$$

A DCT tem, assim como a DFT, uma propriedade de conservação de energia (ou seja, há um análogo do Teorema de Parseval para a DCT). Isso pode ser observado escrevendo a transformada como uma transformação linear. Para isso, considere um sinal $x[n], 0 \leq n \leq N - 1$ com N amostras e defina a DCT2 desse sinal, $X[k]$ como

$$X[k] = \sum_{n=0}^{N-1} x[n] a_k[n],\ 0 \leq k \leq N - 1.$$

Esta relação pode ser escrita na forma matricial definindo um vetor $\boldsymbol{x}$ no $\mathbb{R}^N$ para representar o sinal, uma matriz $\boldsymbol{A}$ cujas colunas sejam os elementos $a_k[n], 0 \leq n \leq N - 1$, e um vetor $\boldsymbol{X}$ para representar a transformada:

$$\boldsymbol{x} = \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} a_0[0] & a_1[1] & \dots & a_0[N-1] \\ a_1[0] & a_1[1] & \dots & a_1[N-1] \\ \vdots & \vdots & \ddots & \vdots \\ a_{N-1}[0] & a_{N-1}[1] & \dots & a_{N-1}[N-1] \end{bmatrix}, \quad \boldsymbol{X} = \begin{bmatrix} X[0] \\ X[1] \\ \vdots \\ X[N-1] \end{bmatrix}.$$

Então verifica-se que vale

$$\boldsymbol{X} = \boldsymbol{A}\boldsymbol{x}.$$

A matriz $\boldsymbol{A}$ definida desta forma é tal que $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{I}$, ou seja, a inversa de $\boldsymbol{A}$ é a sua transposta $\boldsymbol{A}^T$: $\boldsymbol{A}^T = \boldsymbol{A}^{-1}$. Matrizes com essa propriedade são chamadas *matrizes ortogonais*. Uma propriedade semelhante é válida para a matriz correspondente à DFT, apenas com uma diferença de escala no caso da definição mais comumente usada em processamento de sinais. De fato, definindo $\boldsymbol{F}$ como a matriz com elementos

$$[\boldsymbol{F}]_{nk} = e^{-j\frac{2\pi}{N}nk},$$

segue que a DFT da sequência representada pelo vetor $\boldsymbol{x}$ é $\boldsymbol{F}\boldsymbol{x}$, e vale $\boldsymbol{F}^H\boldsymbol{F} = N\boldsymbol{I}$, em que o sobrescrito $(\cdot)^H$ representa transposta conjugada, ou seja, $\boldsymbol{F}^H = \left(\boldsymbol{F}^T\right)^*$.

Como a matriz $\boldsymbol{A}$ é ortogonal, a antitransformada do cosseno é simplesmente

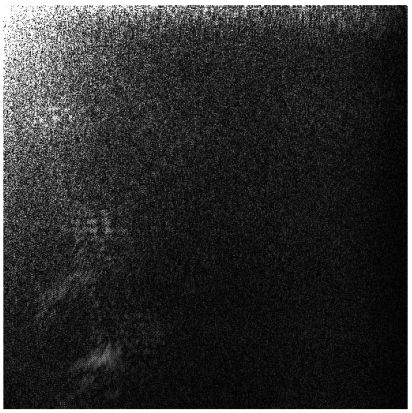
$$\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{X}.$$

Para o caso de imagens, definindo a matriz $\boldsymbol{h}$ com elementos $[\boldsymbol{h}]_{n_1n_2} = h[n_1, n_2]$, e a matriz transformada $[\boldsymbol{H}]_{k_1,k_2} = H[k_1, k_2]$, a expressão da DCT (5) pode ser escrita como

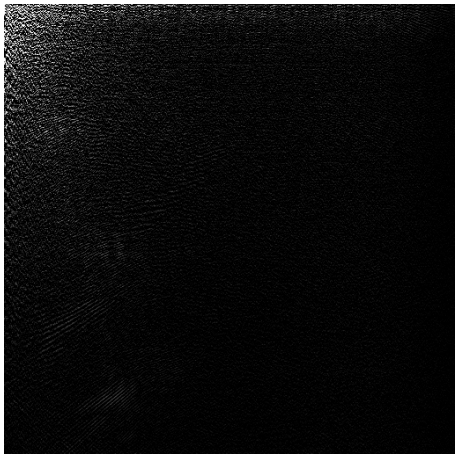
$$\boldsymbol{H} = \boldsymbol{A}\boldsymbol{h}\boldsymbol{A}^T.$$

Considere a imagem da figura 1a, e sua DCT, representada nas figuras 1b–1d. Cada uma dessas últimas representa o valor absoluto da DCT da imagem da figura 1, com o valor 0 representado como preto, e valores acima de um certo limiar representados como branco. Os limiares são: 50 (figura 1b), 100 (figura 1c) e 500 (figura 1d).

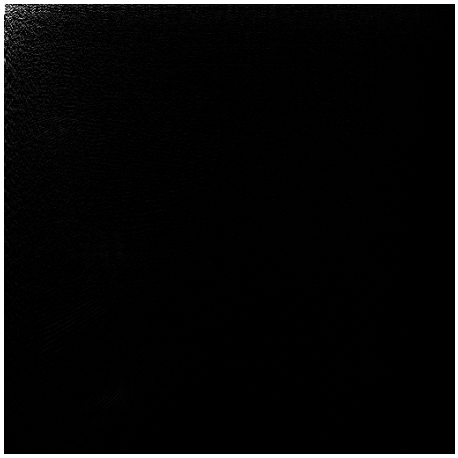
(a) Imagem "goldhill", original.



(b) Valor absoluto da DCT da imagem "goldhill" em escala de cinza. Pontos com valores maiores ou iguais a 50 são mostrados em branco, os demais em preto. O valor DC ($H[0,0]$) corresponde ao ponto superior à esquerda.



(c) Valor absoluto da DCT da imagem "goldhill" em escala de cinza. Pontos com valores maiores ou iguais a 100 são mostrados em branco.



(d) Valor absoluto da DCT da imagem "goldhill" em escala de cinza. Pontos com valores maiores ou iguais a 500 são mostrados em branco.

Figura 1: Concentração dos valores da DCT de uma imagem.

A maior parte da energia da DCT está portanto concentrada em frequências baixas. Podemos verificar isso fazendo um gráfico da energia acumulada nos coeficientes da DCT, ordenados do maior para o menor, como visto na figura 2. Note que 10% dos coeficientes já respondem por mais de 99,5% da energia. Isto significa que imagens obtidas mantendo-se

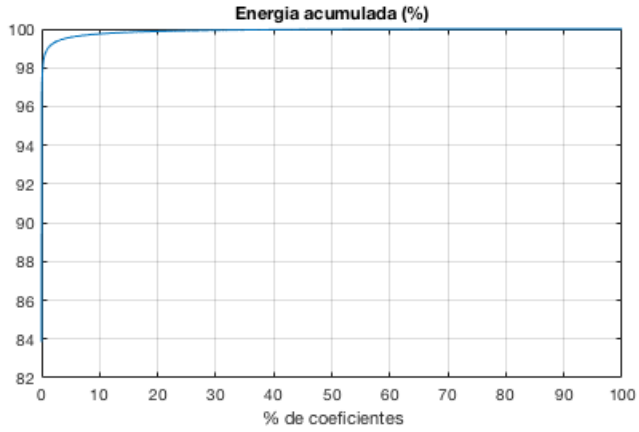


Figura 2: Energia acumulada dos coeficientes ordenados da DCT, em % do valor total.

Figura 3: Imagem obtida mantendo-se apenas os 10% maiores coeficientes da DCT.

apenas os poucos coeficientes responsáveis pela maior parte da energia têm boa qualidade. Por exemplo, a figura 3 é a DCT inversa mantendo-se apenas os 10% maiores coeficientes da DCT. A PSNR resultante é de 32,36 dB.

No padrão JPEG a DCT é calculada sobre blocos 8×8 da imagem original. Cada coeficiente da DCT de cada bloco é então quantizado com um passo de quantização diferente, obtido de uma tabela como a seguir [1]:

$$\boldsymbol{Q} = \begin{bmatrix} 8 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}. \tag{7}$$

Assim, o coeficiente DC da DCT é quantizado com passo de quantização 8, o coeficiente $(1, 2)$ é quantizado com passo 11, o coeficiente $(2, 1)$ com passo 12, e assim por diante. Repare que os passos de quantização vão aumentando para frequências maiores, de modo que coeficientes relativos a frequências altas são quantizados com menos bits para uma mesma faixa dinâmica.

É possível usar-se passos de quantização maiores (usando $k\boldsymbol{Q}$ com $k > 1$), de maneira a aumentar a taxa de compressão da imagem (mas com uma qualidade pior).

Se as dimensões da imagem não forem múltiplas de $(8, 8)$, a imagem é aumentada, geralmente espelhando-se as bordas para evitar artefatos devidos a bordas abruptas.

4 Codificação dos coeficientes quantizados

O último passo do codificador JPEG é transformar cada coeficiente em um código, procurando reduzir o número total de bits. No padrão JPEG usa-se códigos de Huffman (ver a apostila [3]) para gerar a representação final da imagem.

A figura 4 mostra um esquema geral do codificador JPEG. A imagem original (por

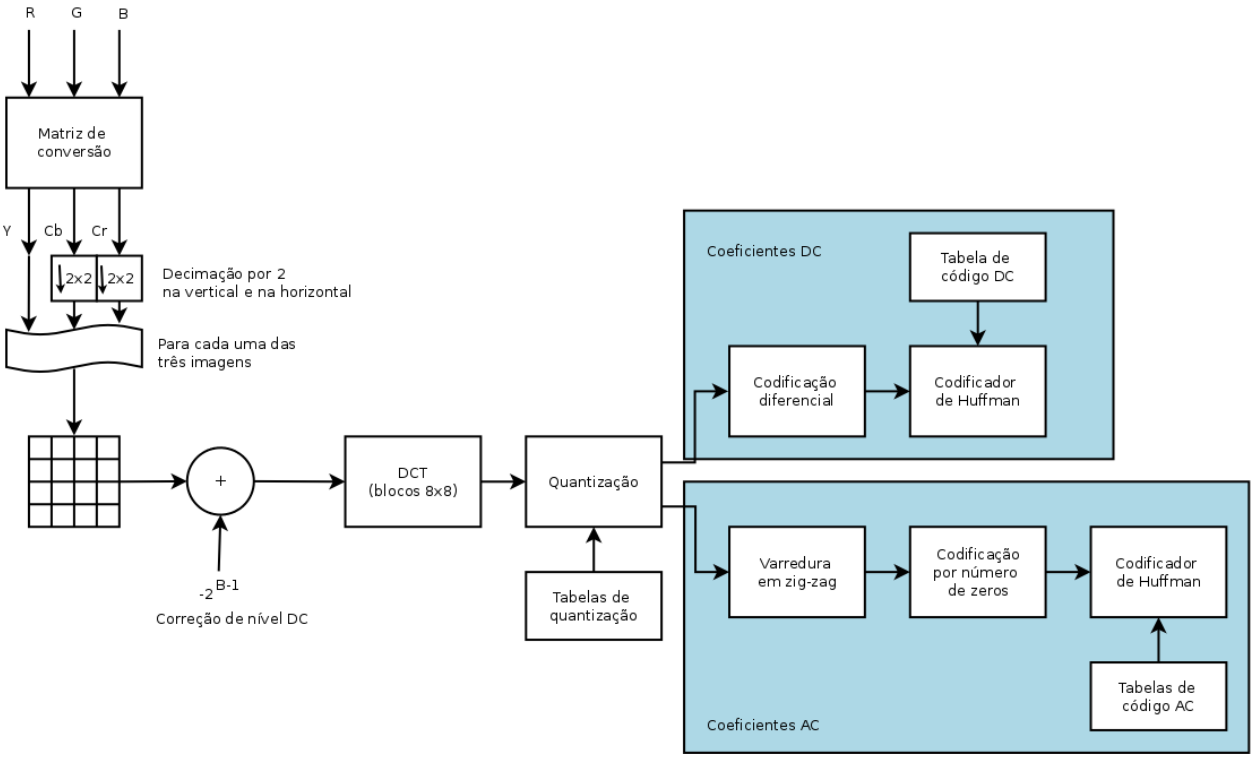


Figura 4: Esquema completo do codificador JPEG (adaptado de [1]).

exemplo, em padrão RGB) é convertida para YCbCr, e os canais de crominância são subamostrados na horizontal e na vertical por um fator de 2. Cada uma das imagens resultantes é processada separadamente como a seguir.

Primeiramente é subtraído um nível DC igual a 2^{B-1} das imagens, de maneira a fazer os sinais ficarem entre -2^{B-1} e $2^{B-1} - 1$. No caso de imagens com 8 bits por pixel, subtrai-se 128. Em seguida, se as dimensões da imagem não forem múltiplas de 8, a imagem é aumentada, espelhando-se as laterais.

É então calculada a DCT de cada bloco 8×8 da imagem resultante, e cada coeficiente é quantizado de acordo com a tabela de quantização (7) e um fator k desejado. O tratamento a seguir é diferente para os coeficientes DC (elementos $(0, 0)$ de cada DCT) e para os coeficientes AC (demais elementos de cada DCT).

Como os coeficientes DC são muito correlacionados entre si, primeiramente é usado um filtro de descorrelação *apenas nos coeficientes DC* com resposta ao impulso

$$h[n_1, n_2] = \begin{bmatrix} -0{,}25 & -0{,}25 & -0{,}25 \\ -0{,}25 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Repare que este é um filtro passa-altas (o ganho DC é nulo), de maneira a codificar essencialmente a diferença entre os coeficientes obtidos em blocos próximos. Repare que é calculada a diferença com relação aos blocos circundantes que são visitados anteriormente na codificação e na decodificação. Depois de descorrelacionados, os coeficientes DC são codificados com um código de Huffman.

Os coeficientes AC são tratados de maneira um pouco diferente. Depois de quantizados, os coeficientes em cada bloco são percorridos por uma varredura em zig-zag (como mostrado na figura 5), para aproximar uma ordem de frequência crescente. Isto é feito porque, como

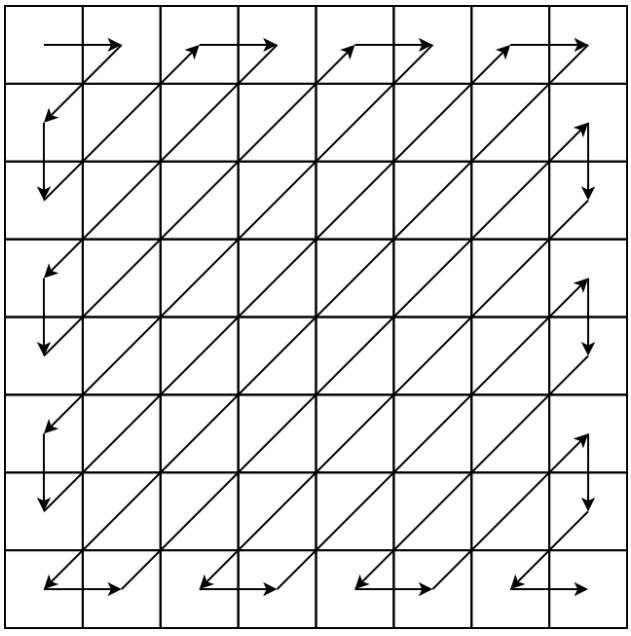


Figura 5: Varredura em zig-zag.

há vários coeficientes nulos, o codificador na verdade codifica apenas os valores não-nulos, e o número de coeficientes nulos entre cada valor não-nulo.

Nesta experiência vamos testar algumas das ideias do codificador JPEG, mas não implementar o padrão em sua totalidade.

5 Parte experimental

Vamos a seguir testar diversos aspectos do padrão JPEG: a sub-amostragem do sinal de crominância, o espelhamento para que as dimensões da imagem sejam múltiplas de 8, a quantização da DCT calculada por blocos com diferentes valores de k, a correlação dos coeficientes DC, e a taxa aproximada do codificador de Huffman. Não serão testados aspectos como a construção do codificador propriamente dito, da construção do código, nem o codificador por número de zeros.

5.1 Sub-amostragem do sinal de crominância

Vamos começar testando a compressão do sinal de crominância. Use a imagem `cat.png` para os testes (você pode também experimentar com outras imagens, mas forneça no relatório os resultados para `cat.png` para comparação entre os resultados dos alunos). A imagem pode ser carregada com o comando `gato=imread('cat.png')` em Matlab. Em Julia, use `using Images, FileIO` e `gato=load("cat.png"`.

Note que em Matlab, os pixels das imagens são números inteiros sem sinal entre 0 e 255 (tipo `uint8`. Em Julia, os pixels são números em ponto fixo entre 0.0 e 1.0 (representados por números de 8 bits sem sinal). Então, em Matlab é necessário converter os números para `double` para fazer as operações, e depois converter novamente para `uint8` para ver as imagens, enquanto que em Julia essa conversão não é necessária.

Por outro lado, uma imagem colorida em Matlab é representada por um arranjo de três dimensões, em que `imagem(:,:,1)` corresponde ao canal R, e similarmente para os canais G e B, nesta ordem. Em Julia, uma imagem colorida é uma matriz de pixels, em que cada pixel tem três componentes. Assim, `red.(imagem)` retorna uma matriz com os valores do canal vermelho da imagem, e assim por diante.

Realize as seguintes tarefas:

1. A variável `gato` tem $733 \times 490 \times 3$ elementos. Os sinais R, G e B são correspondentes a `gato(:, :, c)`, com `c` igual a 1, 2 ou 3, respectivamente, em Matlab. Em Julia, os canais são recuperados com `red.(gato)`, `green.(gato)` e `blue.(gato)`. Gere os sinais $\boldsymbol{Y}, \boldsymbol{C}_b, \boldsymbol{C}_r$ usando a transformação (3).

2. Sub-amostre os sinais $\boldsymbol{C}_b$ e $\boldsymbol{C}_r$ por um fator de 2 nas direções vertical e horizontal. Calcule a taxa de compressão obtida com esta transformação, considerando que as imagens tanto no padrão RGB quanto no padrão YCbCr são armazenadas com o mesmo número de bits em cada canal.

3. Recupere os sinais RGB a partir de $\boldsymbol{Y}, \boldsymbol{C}_b, \boldsymbol{C}_r$, usando a inversa de (3) e usando usando interpolação linear, usando o filtro com resposta ao impulso

$$h = \begin{bmatrix} 0{,}25 & 0{,}5 & 0{,}25 \\ 0{,}5 & 1 & 0{,}5 \\ 0{,}25 & 0{,}5 & 0{,}25 \end{bmatrix}$$

e a função `filter2`: `saida=filter2(h,entrada);` em Matlab. Em Julia, a filtragem 2D é feita com o comando `imfilter`, e a sintaxe `imfilter(entrada, filtro, Fill(0, filtro)`. Verifique num diagrama que este filtro realmente aplica interpolação linear para aproximar as linhas e colunas eliminadas da imagem.

O `image toolbox` do Matlab tem a função `imresize` que faz a mesma função com outras opções de interpolação, mas use a interpolação linear como descrito acima.

4. Compare a imagem recuperada com a imagem original usando o comando `imshow` (Matlab). Calcule a PSNR da imagem recuperada.

5.2 Quantização da DCT

Vamos agora testar o uso da DCT e a quantização dos seus coeficientes.

1. Acrescente um número mínimo de linhas e colunas aos canais $\boldsymbol{Y}, \boldsymbol{C}_b, \boldsymbol{C}_r$ de modo que as dimensões das imagens sejam múltiplas de 8, acrescentando à bordas direita e inferior da imagem um número de colunas e linhas adequado. As linhas e colunas acrescentadas devem conter uma imagem espelhada das últimas linhas e colunas da imagem original, para evitar efeitos de borda da DCT. Isto é equivalente em Matlab a usar o comando `padarray(imagem, [px py], 'symmetric', 'post');`, em que `px` e `py` são os números de pixels a acrescentar em cada direção. Em Julia, use `padarray(imagem, Pad(:symmetric, px, py))` (note que a saída desejada está nas coordenadas `[1:end, 1:end]`) — em Julia, a imagem tem linhas e colunas acrescentadas em todos os lados, e a saída do comando `padarray` é acessada com índices de `[-px:nx+px, -py:ny+py]`. Aplique o *padding* aos sinais de crominância antes de decimá-los, e decime o resultado como no item anterior.

2. Aplique a DCT a blocos 8×8 de cada canal, subtraindo antes 128 das imagens em Matlab, ou 0.5 em Julia. Para fazer isso, defina uma variável com zeros do mesmo tamanho da imagem, e use um laço `for` para calcular a DCT de cada bloco e colocar o resultado na variável de saída. Note que no `image toolbox` do Matlab poderia ser usado o comando `blkproc`

   ```
   Ydct = blkproc(Y-128, [8 8], @dct2);
   ```

   para cada canal.

   Em Julia a DCT de duas dimensões é obtida com o comando `dct(bloco)`, e é possível usar matrizes de matrizes para calcular as DCTs dos blocos de maneira conveniente (mas é necessário escrever uma função para converter matrizes de matrizes para uma matriz única para recuperar a imagem no final).

3. Quantize os coeficientes da DCT usando $k\boldsymbol{Q}$ como passo de quantização, com $k = 1, 2, 3$ (em Matlab, você pode usar `blkproc`). Você pode quantizar uma variável `x` com um passo `q` usando o comando `xq = round(x ./ q)`. Em Julia, a quantização deve ser

feita usando `xq = round(255x ./ q) / 255`, já que os pixels são interpretados como números fracionários.

4. Recupere a imagem quantizada, invertendo o quantizador (`xrec = xq .* q`), aplicando a DCT inversa (comando `idct2`), somando novamente 128 ao resultado, e interpolando os sinais de crominância, como antes. Lembre-se de cortar a parte espelhada antes de desenhar a imagem resultante e de calcular a PSNR obtida para cada valor de k.

5. Calcule o número de coeficientes nulos e a razão entre o número de pixels original e o número de coeficientes nulos da DCT (considere todos os canais). Note que esta não é a taxa de compressão da imagem.

5.3 Cálculo da entropia do sinal

Para estimar a taxa de compressão da imagem, vamos calcular a entropia da imagem `cat.png`. Para isto, assuma que os valores possíveis para os coeficientes DC estejam no intervalo $-2047 : 2047$, e os valores possíveis para os coeficientes AC estejam no intervalo $-1023 : 1023$.

1. Calcule a probabilidade de cada valor de código ocorrer para a imagem `cat.png` (usando os três canais, $\boldsymbol{Y}, \boldsymbol{C}_b, \boldsymbol{C}_r$ e as DCTs quantizadas). Dica: em Matlab o comando `sum(sum(X==1))` calcula o número de elementos da matrix `X` que são iguais a 1, em Julia, `sum(X .== 1)`. Faça o cálculo separadamente para os coeficientes DC e para os coeficientes AC.

2. Calcule a entropia do sinal resultante pela definição [3]:

$$H(S) = -\sum_{k=1}^{K} \log_2(p_k) p_k \text{ (bits)}, \tag{8}$$

em que p_k é a probabilidade do sinal assumir o valor s_k.

3. Como a entropia é o menor comprimento médio para qualquer código que represente uma fonte sem perdas, estime o tamanho necessário para um arquivo JPEG representando a imagem `cat.png`.

4. Verifique que os coeficientes DC são correlacionados desenhando um gráfico que desenhe (para o canal $\boldsymbol{Y}$) o valor do coeficiente DC de um bloco em função do valor do coeficiente DC no bloco seguinte. Veja que aparece uma clara correlação entre os coeficientes. Repita para um coeficiente AC qualquer e repare que (em geral) não é discernível nenhuma correlação elevada. Faça os gráficos com os valores não quantizados da DCT.

Referências

[1] T. Dutoit e F. Marqués. *Applied Signal Processing*. Springer, 2009.

[2] C. Poynton. Frequently asked questions about color. http://poynton.ca/Poynton-color.html, out. 2018.

[3] M. T. M. Silva. Uma introdução à teoria da informação. Apostila da disciplina PSI3531, 2018.