3º Ano – Licenciatura em Engenharia Informática

Multimédia

Trabalho Prático nº1

Compressão de Imagem

· U C ·

3º Ano – Licenciatura em Engenharia Informática

Multimédia

Trabalho Prático nº1

Compressão de Imagem

Trabalho realizado por:

André Caseiro Graça, 2019215067

João A. da Silva Melo, 2019216747

Miguel A. G. de Almeida Faria, 2019216809

· U C ·

Índice

Introdução

Este trabalho prático foi realizado no âmbito da disciplina de Multimédia, com o objetivo de adquirir conhecimentos acerca de questões fundamentais de compressão de imagem. Com recurso à linguagem *Python* e a várias bibliotecas, foram implementados mecanismos utilizados na compressão através do *codec jpeg* e retiradas conclusões acerca dos dados obtidos.

Exercício 1 – Compressão de imagens bmp no formato jpeg utilizando um editor de imagem

Para realizar as compressões para formato *jpeg* das imagens fornecidas, utilizamos o programa GIMP, onde conseguimos testar com diferentes fatores de qualidade de acordo com a convenção: baixa (25), média (50) e alta (75).

As tabelas seguintes apresentam os dados recolhidos, relativamente a tamanho, taxa de compressão e qualidade subjetiva, para cada uma das imagens:

barn_mountains.bmp	Qualidade			
	Original	Baixa (25)	Média (50)	Alta (75)
Tamanho	348 KB	13.5 KB	21.5 KB	33.0 KB
Taxa de Compressão	1 : 1	1 : 26	1 : 16	1 : 11
Qualidade Subjetiva	Original	Pouca definição	Algum desfoque	Pouca diferença

Tabela 1 – Dados de compressão da imagem "barn_mountains.bmp"

logo.bmp	Qualidade			
	Original	Baixa (25)	Média (50)	Alta (75)
Tamanho	411 KB	6.18 KB	7.60 KB	9.43 KB
Taxa de Compressão	1 : 1	1 : 67	1 : 54	1 : 44
Qualidade Subjetiva	Original	Manchas visíveis	Pouca definição	Bom

Tabela 2 – Dados de compressão da imagem "logo.bmp"

peppers.bmp	Qualidade			
	Original	Baixa (25)	Média (50)	Alta (75)
Tamanho	576 KB	13.30 KB	20.3 KB	30.5 KB
Taxa de Compressão	1 : 1	1 : 43	1 : 28	1 : 19
Qualidade Subjetiva	Original	Muita pixelização	Pouca definição	Pouca diferença

Tabela 3 – Dados de compressão da imagem "peppers.bmp"

Exercício 1.4

Em todas as imagens, independentemente da qualidade de compressão utilizada, consegue-se reduzir de forma bastante significativa o seu tamanho. Verifica-se que a imagem com mais redundância, neste caso a imagem "logo.bmp", é aquela onde se obtém taxas de compressão maiores.

De uma maneira geral, para todas as imagens, podemos constatar que utilizando a qualidade alta para comprimir as imagens, o resultado vai de encontro ao esperado, apresentando poucas diferenças relativamente à imagem original.

Verifica-se também que à medida que a qualidade diminui, o ruído presente na imagem aumenta, principalmente em zonas da imagem em que existe pouca variação de cor. Um caso em que esta situação está bem evidente é que usando qualidade baixa na imagem "logo.bmp" (que possui pouca variação de cor em certas zonas, mas grande contraste entre elas) a deterioração da imagem é bem mais percetível do que na imagem "barn_mountains.bmp" utilizando a mesma qualidade de compressão (onde existe mais detalhes e variação de cores).

Utilizando a qualidade de compressão média, podemos concluir que existe um melhor equilíbrio entre taxa de compressão e perda de qualidade da imagem, correspondendo às espectativas.

Exercício 2 – Criação de funções *encoder* e *decoder*

De maneira a encapsular as funções que permitem realizar a compressão e descompressão, foram criadas as funções *encoder* e *decoder*:

- encoder(image, quality)
  - Parâmetros:
    - image - imagem que se pretende comprimir;
    - quality - qualidade que se pretende aplicar.

  - Retorno:
    - img – array contendo a informação da imagem original;
    - (Y_DCT8, Cb_DCT8, Cr_DCT8) – tuplo contendo a informação dos canais YCbCr da imagem comprimida, através de DCT em blocos, quantização e DPCM
    - nl – número de linhas da imagem original;
    - nc – número de colunas da imagem original;
    - samplingType – tipo de *downsampling* aplicado;
    - blocksize – dimensão dos blocos aplicados na realização da DCT;

        - quantizationQuality – fator de qualidade aplicado na quantização (ou se não foi aplicada, caso seja usado valores inválidos);
        - DPCM - variável que indica se foi aplicada codificação DPCM ou não

- decoder(imageEncoded, nl, nc, samplingType, blockSize, quantizationQuality, DPCM)
    - Parâmetros:
        - imgEncoded – tuplo contendo os canais da imagem derivados da compressão;
        - nl – número de linhas da imagem original;
        - nc – número de colunas da imagem original;
        - samplingType – tipo de *downsampling* aplicado;
        - blocksize – dimensão dos blocos aplicados na realização da DCT;
        - quantizationQuality – fator de qualidade aplicado na quantização (ou se não foi aplicada, caso seja usado valores inválidos);
        - DPCM - variável que indica se foi aplicada codificação DPCM ou não.
    - Retorno:
        - img – array contendo a informação imagem reconstruída;

De notar que a função *encoder* possui alguns métodos que não intervêm diretamente na compressão de uma imagem, mas que ajudam a visualizar alguns dos exercícios pedidos na ficha (nomeadamente nos casos em que é pedida a visualização de certas etapas da compressão, bastando nesses casos apenas alterar o valor do parâmetro "show" para *true*).

Exercício 3 – Visualização de imagem representada pelo modelo de cor RGB

Para facilitar a resolução de problemas seguintes, foram criadas funções (e, caso necessário, as suas inversas) para separar os canais de uma imagem, quer esteja em RGB ou em YCbCr, e para visualizar um determinado canal usando um *colormap* definido pelo utilizador. Além disso foi implementada uma função que permite visualizar os canais RGB de uma imagem com os *colormaps* respetivos: vermelho, verde e azul.

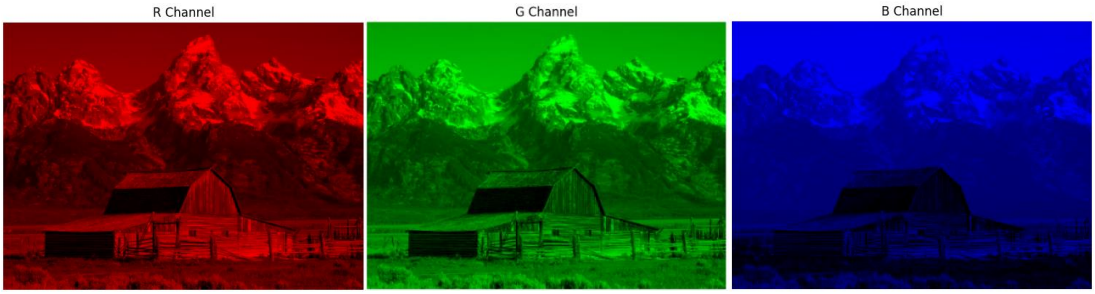

Figura 1 – Canais R, G e B da imagem "barn_mountains.bmp" com os respetivos *colormaps*

Exercício 4 – Pré-processamento da imagem: padding

De maneira a efetuar o *padding* de uma imagem, foi desenvolvida uma função (e a sua inversa) que permite replicar a última linha e/ou coluna, caso a sua dimensão não obedeça a um tamanho definido pelo utilizador, sendo 16 o valor mais comum e utilizado neste trabalho com maior importância.

Exercício 5 – Conversão para o modelo cor YCbCr

Uma vez que a compressão é mais fácil de realizar caso a imagem esteja no modelo YCbCr, foi criada uma função que permite converter uma imagem RGB neste modelo, bem como a função inversa. Para observar os canais Y, Cb e Cr, foi também elaborada uma função onde se utiliza um *colormap* cinzento para a sua visualização.

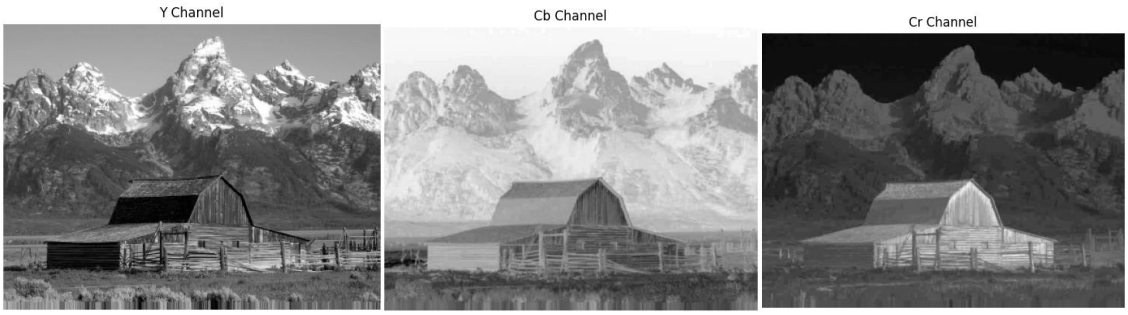

Figura 2 – Canais Y, Cb e Cr da imagem "barn_mountains.bmp" com o *colormap* cinzento

Exercício 5.4

Comparando o canal Y com os restantes canais do modelo YCbCr, conseguimos verificar que é nesse canal onde se encontra maior detalhe da imagem, sendo essa umas das razões pela qual não deve ser aplicado *downsampling* nesse canal.

No caso dos canais R, G e B, observa-se que o detalhe da imagem se encontra distribuído de igual forma entre eles, embora quando comparado com o canal Y esse continue também a ser menor.

Exercício 6 – Sub-amostragem

Para realizar a sub-amostragem de uma imagem, foram implementadas funções que permitem ao utilizador realizar *downsampling* e *upsampling* de 4:2:2 e 4:2:0 nos canais Cb e Cr, bem como uma função para visualizar os canais após estas operações.

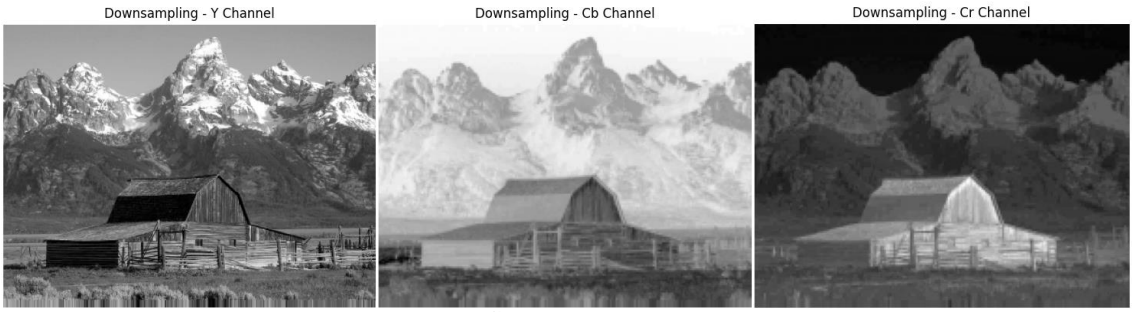

Figura 3 – Canais Y, Cb e Cr com aplicação de *downsampling* da imagem "barn_mountains.bmp"

Exercício 6.3

Após a utilização de *downsampling* com interpolação verificou-se que a destrutividade, apesar de ainda existente, era significativamente menor, indo de encontro aquilo que o algoritmo realiza.

Em termos de compressão, aplicando um *downsampling* de 4:2:2, os canais Cb e Cr ficarão reduzidos a metade, perdendo uma a cada 2 colunas da sua matriz. Assim sendo, será seguro dizer que a taxa de compressão da imagem será o canal Y (1/3), com metade dos canais Cb e Cr (2/6 = 1/3), concluindo-se então que a taxa de compressão com este *downsampling* será de 2:3 do tamanho original da imagem.

Sabendo que o *downsampling* de 4:2:0 se aplica de igual forma às colunas que o *downsampling* de 4:2:2, com a adição do mesmo algoritmo às linhas dos canais Cb e Cr, a taxa de compressão em cada um destes canais será de 1/4. Assim, ao aplicar um *downsampling* de 4:2:0, a taxa de compressão será de: 1/3 + 1/3 * 1/4 * 2 = 1/2. Em suma, a taxa de compressão com este *downsampling* será de 1:2 do tamanho original da imagem.

	Tipo de *Downsampling*		
	Original	4:2:0	4:2:2
Tamanho	N	1/2*N	2/3*N

Tabela 4 – Tamanhos espectáveis usando os diferentes tipos de *downsampling*

Sabendo que após a aplicação do *downsampling* numa imagem há sempre destruição, para reduzir o impacto desta, aplicámos o método com interpolação de cor, em que apesar de ainda haver alguma perda, esta é bastante menor, especialmente nas imagens "barn_mountains.bmp" e "peppers.bmp", em que há uma maior variedade de cor. Na imagem "logo.bmp", possuindo esta um número reduzido de cores, a interpolação não teve um efeito tão significativo.

Exercício 7 – Transformada de Cosseno Discreta (DCT)

Com o objetivo de conseguir aplicar a DCT, a sua inversa ou a DCT segundo uma transformação logarítmica a um determinado canal ou bloco, foi implementada uma função recorrendo aos comandos scipy.fftpack.dct e scipy.fftpack.idct.

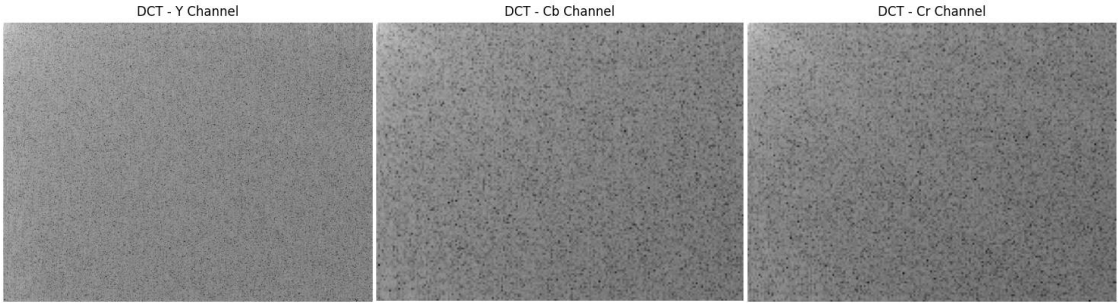

Figura 4 – Canais Y, Cb e Cr com aplicação de DCT logarítmica da imagem "barn_mountains.bmp"

Exercício 7.1.3

Ao observar o resultado da aplicação da DCT aos canais Y, Cb e Cr das imagens, é possível identificar quais as que terão uma melhor compressão devido à concentração das baixas frequências espaciais no canto superior esquerdo após a aplicação desta transformada. Assim, verifica-se que nas imagens "barn_mountains.bmp" e "peppers.bmp", há uma maior concentração das baixas frequências no canto superior esquerdo e na imagem "logo.bmp" há uma maior distribuição destas. É possível observar nas seguintes imagens as diferentes distribuições das baixas frequências no canal Y das imagens dadas.

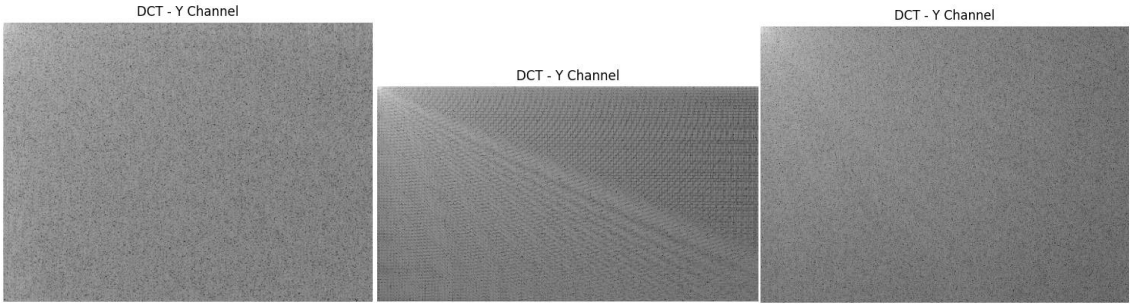

Figura 5 – Canal Y com aplicação de DCT de todas as imagens ("barn_mountains", "logo", "peppers")

Exercício 7.2.3

As seguintes imagens são a representação do canal Y das imagens fornecidas. Como se consegue verificar, há facilidade em distinguir os diferentes blocos e a distribuição das baixas frequências é uniforme, sendo que em cada bloco existe uma concentração destas no seu canto superior esquerdo, resultando numa boa compressão alcançada por bloco. A imagem "logo.bmp" deixa de ter estas frequências distribuídas pela imagem toda, havendo apenas uma maior distribuição de frequências nos blocos de fronteira entre cores.

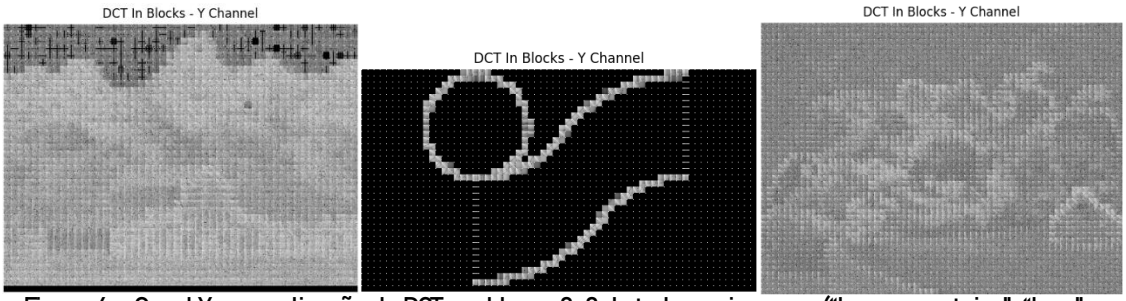

Figura 6 – Canal Y com aplicação de DCT em blocos 8x8 de todas as imagens ("barn_mountains", "logo", "peppers")

Exercício 7.3.2

Os resultados obtidos para os canais Y das imagens foram os seguintes:

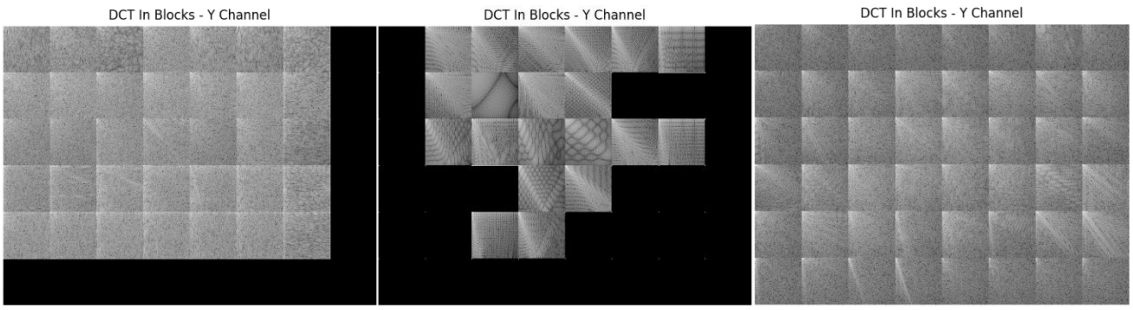

Figura 7 – Canal Y com aplicação de DCT em blocos 64x64 de todas as imagens ("barn_mountains", "logo", "peppers")

Consegue-se observar que existe uma maior distribuição das baixas frequências em cada bloco, deixando de haver a grande concentração no canto superior esquerdo de cada um deles. Consequentemente a compressão dos blocos irá ser pior.

Exercício 8 – Quantização

Para a aplicação da quantização e de-quantização na imagem, foram utilizadas as matrizes de quantização propostas no standard. Para além disso, a função permite ao utilizador escolher o fator de qualidade que pretender.

Exercício 8.1

Como foi verificado no Exercício 1, quanto menor é o fator de qualidade, maior é a taxa de compressão. Assim, ao aplicar um menor fator de qualidade a quantização irá aumentar, sendo que esta vai remover as informações de elevada frequência. Também a destrutividade irá aumentar e consequentemente, a taxa de compressão. Além disso, as diferentes imagens são afetadas de maneira diferente, a quantização está dependente da quantidade e dos valores das diferenças de cor.

Exercício 8.2

Os canais Y, Cb e Cr estão agora divididos em blocos e cada bloco está fragmentado em partes de diferentes frequências. Há menos detalhe e menos variedade de cor. Caso haja um fator de qualidade menor a imagem estará mais destruída pela quantização. O canal Y é o que tem mais detalhe e é o que sofre menos quantização.

Exercício 9 – Codificação DPCM dos coeficientes DC

Por fim, foi elaborada uma função que permite realizar a codificação dos coeficientes DC de cada bloco, bem como a sua inversa.

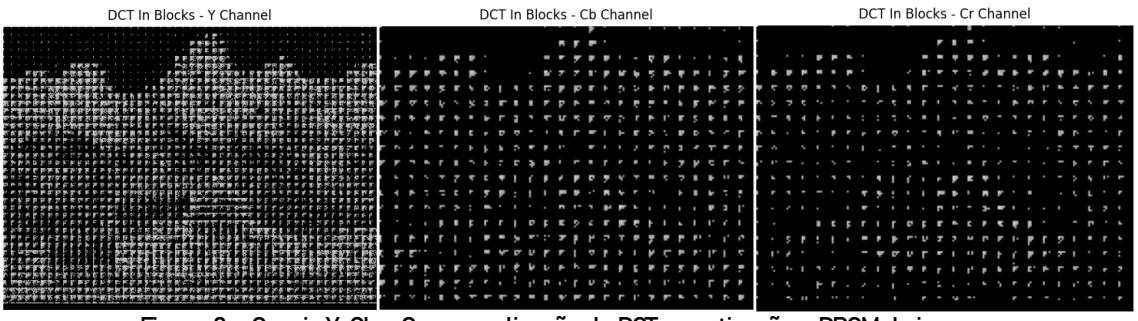

Figura 8 – Canais Y, Cb e Cr com aplicação de DCT, quantização e DPCM da imagem "barn_mountains.bmp"

Exercício 9.3

Verifica-se que há alterações nos cantos superiores esquerdos dos blocos da imagem. Isto acontece porque o DPCM retira ao primeiro valor do bloco atual, o primeiro valor original do bloco anterior, o que diminui a frequência nesse ponto. O bloco fica então com uma gama de valores menor, o que aumenta a compressibilidade.

Exercício 10 – Codificação e descodificação end-to-end

Ao aplicar a descodificação da informação que sofreu todos os processos referidos, obtém-se a seguinte imagem:

Figura 9 – Imagem "barn_mountains.bmp" descodificada

As tabelas apresentadas a seguir possuem os valores das diferentes métricas de distorção nas imagens fornecidas, para os vários fatores de qualidade:

barn_mountains.bmp	Qualidade				
	10	25	50	75	100
MSE	727.662	415.814	278.080	168.821	26.253
RMSE	26.975	20.392	16.676	12.993	5.124
SNR	18.567	20.997	22.745	24.912	32.995
PSNR	19.512	21.942	23.689	25.857	33.939

Tabela 5 – Valores das métricas de distorção para a imagem "barn_mountains.bmp"

logo.bmp	Qualidade				
	10	25	50	75	100
MSE	173.331	79.692	53.222	32.070	13.393
RMSE	13.166	8.927	7.295	5.663	3.660
SNR	28.943	32.318	34.071	36.271	40.063
PSNR	25.742	29.117	30.870	33.070	36.862

Tabela 6 – Valores das métricas de distorção para a imagem "logo.bmp"

peppers.bmp	Qualidade				
	10	25	50	75	100
MSE	323.414	154.725	101.675	68.749	19.656
RMSE	17.984	12.439	10.083	8.291	4.433
SNR	19.828	23.030	24.854	26.553	31.991
PSNR	23.033	26.235	28.059	29.758	35.196

Tabela 7 – Valores das métricas de distorção para a imagem "peppers.bmp"

Através da análise das tabelas, conclui-se que, em todas as imagens, tanto o valor MSE como o RMSE decrescem à medida que a qualidade aumenta, enquanto os valores de SNR e PSNR aumentam.

Para além disso, é também percetível que nas imagens em que a variação de cor é menos abrupta, "barn_mountains.bmp" e "peppers.bmp", há um maior MSE e consequentemente maior RMSE. Isto é de esperar, sendo que o *jpeg* tem um melhor desempenho em imagens com transições suaves. A destruição apesar de ser bastante mais significativa nos valores de MSE e RMSE, ao nível de uma análise visual subjetiva, com uma maior qualidade, a diferença entre a imagem original e a reconstruída é pequena. Assim, para haver um MSE maior, foram eliminadas as frequências que não têm grande impacto à visão humana (frequências altas), mas que contribuem para uma melhor compressão, sendo esse um dos principais objetivos do *jpeg* nas suas fases destrutivas.

Exercício 10.3

É possível verificar que os resultados obtidos no exercício 1 vão de encontro ao que foi obtido nesta análise do *encoder* e *decoder*. A imagem "logo.bmp", devido à existência de mudanças abruptas de cor e grande redundância, é a que consegue uma maior compressão, porém, também é a que tem uma maior destruição e mais deformação na sua reconstrução, devido ao mau funcionamento do *jpeg* neste tipo de imagens. Verifica-se também que quanto menor a qualidade, maior a destruição e compressão.

Conclusão

Este trabalho permitiu-nos implementar e analisar vários mecanismos utilizados na compressão de imagem através do *codec jpeg*. Dessa forma conseguimos responder ao que era pretendido e alcançar o objetivo do trabalho que era adquirir conhecimentos acerca de questões fundamentais de compressão de imagem.

Como referido anteriormente, concluímos que é ao realizar *downsampling* e quantização que existe maior destruição, sendo estes, consequentemente, os processos que contribuem de maior forma para a compressão das imagens. Por último, verificamos que o trabalho desenvolvido poderá ser complementado por *codecs* de compressão de informação com RLE ou Huffman, atingindo assim o propósito final do *jpeg*.

Bibliografia

[1] PowerPoints fornecidos nas aulas
[2] www.numpy.org