Multimédia

Trabalho Prático nº 1

Compressão de Imagem

Objectivo

Pretende-se que o aluno adquira sensibilidade para as questões fundamentais de **Compressão de Imagem**, em particular através do codec **JPEG**.

Planeamento

Prazo de Entrega:

25 de Março, sexta-feira, 23h59

Esforço extra-aulas previsto: 18h/aluno

Formato de Entrega:

1) Entrega final (código completo + relatório): InforEstudante
2) Notas:
   a) Gerar o ficheiro zip, contendo o pdf do relatório, os ficheiros com o código e outros ficheiros que considere relevantes;
   b) Para evitar erros de submissão no inforestudante, gerar acrescentar a extensão .pdf ao ficheiro (e.g., ficheiro.zip → **ficheiro.zip.pdf**)

Período de execução: 6 aulas práticas laboratoriais

Ritmo de execução esperado para a avaliação contínua:

- Semana 1: alíneas 1 a 5
- Semana 2: alíneas 6 a 7.1
- Semana 3: alíneas 7.2 a 7.3
- Semana 4: alínea 8 e 9
- Semana 5: alínea 10
- Semana 6: correcções e relatório (recomendação: o relatório deverá ser escrito ao longo das 6 semanas)

Trabalho Prático

Implementação e análise de mecanismos utilizados na compressão de imagens através do codec JPEG, usando Python.

1. Compressão de imagens bmp no formato jpeg utilizando um editor de imagem (e.g., GIMP, Adobe Photoshop, etc.).
   1.1. Comprima as imagens fornecidas segundo o codec JPEG, com qualidade alta.
   1.2. Comprima as imagens fornecidas segundo o codec JPEG, com qualidade média.
   1.3. Comprima as imagens fornecidas segundo o codec JPEG, com qualidade baixa.
   1.4. ✍ **Compare os resultados e tire conclusões.**

2. Crie duas funções, *encoder* e *decoder*, para encapsular as funções a desenvolver nas alíenas 3 a 9.

3. Visualização de imagem representada pelo modelo de cor RGB.
   3.1. Leia uma imagem .bmp, e.g., a imagem peppers.bmp.
   3.2. Crie uma função para implementar um *colormap* definido pelo utilizador.
   3.3. Crie uma função que permita visualizar a imagem com um dado *colormap*.
   3.4. Crie uma função para separar a imagem nos seus componentes RGB. Crie também a função inversa.
   3.5. Visualize a imagem e cada um dos canais RGB (com o *colormap* adequado).

4. Pré-processamento da imagem: *padding*.
   4.1. Crie uma função para fazer padding da imagem. Caso a dimensão da imagem não seja múltipla de 16x16, faça padding da mesma, replicando a última linha e a última coluna em conformidade. Crie também a função inversa. Certifique-se de que recupera a imagem com a dimensão original, visualizando-a.

5. Conversão para o modelo cor YCbCr.
   5.1. Crie uma função para converter a imagem do modelo de cor RGB para o modelo de cor YCbCr. Crie também a função inversa (conversão de YCbCr para RGB). Certifique-se de que consegue obter os valores originais de RGB (teste, por exemplo, com o pixel [0, 0]). Nota: na conversão inversa, garanta que R, G e B sejam número inteiros no intervalo {0, 1, …, 255}.
   5.2. Converta a imagem inicial para o modelo de cor YCbCr.
   5.3. Visualize cada um dos canais (com o colormap adequado)
   5.4. ✍ **Compare a imagem de Y com R, G e B e com Cb e Cr. Tire conclusões.**

6. Sub-amostragem.
   6.1. Crie uma função para sub-amostrar os canais Y, Cb, e Cr, segundo as possibilidades definidas pelo codec JPEG, a qual deve devolver Y_d, Cb_d e Cr_d. Crie também a função para efectuar a operação inversa, i.e., upsampling. Certifique-se de que consegue reconstruir com exactidão Y, Cb e Cr.
   6.2. Visualize os canais Y_d, Cb_d e Cr_d com downsampling 4:2:0. Apresente as dimensões das matrizes correspondentes.
   6.3. ✍ **Apresente e analise a taxa de compressão alcançada para as variantes de downsampling 4:2:2 e 4:2:0 (taxa de compressão, destrutividade, etc.)**

7. Transformada de Coseno Discreta (DCT).
   7.1. DCT nos canais completos
      7.1.1. Crie uma função para calcular a DCT de um canal completo. Utilize a função scipy.fftpack.dct. Crie também a função inversa (usando scipy.fftpack.idct). Certifique-se de que consegue obter os valores originais de Y_d, Cb_d e Cr_d. Nota: para uma matriz, X, com duas dimensões, deverá fazer: *X_dct = dct(dct(X, norm="ortho").T, norm="ortho").T*
      7.1.2. Aplique a função desenvolvida a Y_d, Cb_d, Cr_d e visualize as imagens obtidas (Y_dct, Cb_dct, Cr_dct).

         Sugestão: atendendo à gama ampla de valores da DCT, visualize as imagens usando uma transformação logarítmica, e.g., de acordo com o seguinte pseudo-código: *imshow(log(abs(X) + 0.0001))*
      7.1.3. ✍ **Discuta os resultados obtidos em termos de potencial de compressão.**
   7.2. DCT em blocos 8x8
      7.2.1. Usando as mesmas funções para cálculo da DCT, crie uma função que calcule a DCT de um canal completo em blocos BSxBS. Crie também a função inversa (IDCT BSxBS). Certifique-se de que consegue obter os valores originais de Y_d, Cb_d e Cr_d.
      7.2.2. Aplique a função desenvolvida (DCT) a Y_d, Cb_d, Cr_d com blocos 8x8 e visualize as imagens obtidas (Y_DCT8, Cb_DCT8, Cr_DCT8).
      7.2.3. ✍ **Compare os resultados obtidos com os resultados de 7.1.2 e discuta-os em termos de potencial de compressão.**
   7.3. DCT em blocos 64x64.
      7.3.1. Repita 7.2
      7.3.2. ✍ **Compare com os resultados anteriores e tire conclusões.**

8. Quantização.
   8.1. Crie uma função para quantizar os coeficientes da DCT para cada bloco 8x8. Crie também a função inversa.
   8.2. Quantize os coeficientes da DCT, usando os seguintes factores de qualidade: 10, 25, 50, 75 e 100. Visualize as imagens obtidas.

8.3. ✍ **Compare os resultados obtidos com os vários factores de qualidade e discuta-os em termos de potencial de compressão.**
8.4. ✍ **Compare os resultados obtidos com os resultados da alínea 5 e tire conclusões.**

9. Codificação DPCM dos coeficientes DC.
   9.1. Crie uma função para realizar a codificação dos coeficientes DC de cada bloco. Em cada bloco, substitua o valor DC pelo valor da diferença. Crie também a função inversa.
   9.2. Aplique a sua função aos valores da DCT quantizada.
   9.3. ✍ **Analise os resultados e tire conclusões.**

10. Codificação e descodificação end-to-end.
    Nota: As funções criadas na alínea 2 deverão conter, neste momento, todo o código de codificação e descodificação desenvolvido nas alíenas 3 a 9. Note que, após a quantização da DCT, não se pretende, neste trabalho, aplicar os passos de compressão não destrutiva dos coeficientes AC (RLE, Huffman / códigos aritméticos).
    10.1. Codifique as imagens fornecidas com os seguintes parâmetros de qualidade: 10, 25, 50, 75 e 100
    10.2. ✍ **Visualize as imagens descodificadas. Visualize também a imagem das diferenças entre o canal Y de cada uma das imagens originais e da imagem descodificada respectiva para cada um dos factores de qualidade testados. Calcule as várias métricas de distorção (MSE, RMSE, SNR e PSNR) para cada uma das imagens e factores de qualidade. Tire conclusões.**
    10.3. Volte a analisar o ponto 1, de forma a validar/complementar as conclusões tiradas nesse ponto.