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:

15 de Março, sexta-feira, 23h59

Esforço extra-aulas previsto: 18h/aluno

Formato de Entrega:

1) Entrega final (código completo + relatório): InforEstudante
2) Notas: Gerar o ficheiro zip, contendo o pdf do relatório, os ficheiros com o código e outros ficheiros que considere relevantes.

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 e 7
- Semana 3: alíneas 8 e 9
- Semana 4: alínea 10
- Semana 5 e 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).
   1.1. Comprima as imagens fornecidas segundo o codec JPEG, com qualidade alta (Q=75).
   1.2. Comprima as imagens fornecidas segundo o codec JPEG, com qualidade média (Q=50).
   1.3. Comprima as imagens fornecidas segundo o codec JPEG, com qualidade baixa (Q=25).
   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. Nota: a ordem da chamada de funções no decoder deve ser inversa da do encoder, dado que corresponderá à inversão da cofificação realizada.

3. **Visualização** de imagem representada pelo modelo de cor RGB.
   3.1. Leia uma imagem .bmp, e.g., a imagem airport.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. *Encoder*: Crie uma função para separar a imagem nos seus componentes RGB.
   3.5. *Decoder*: Crie também a função inversa (que combine os 3 componentes RGB).
   3.6. Visualize a imagem e cada um dos canais RGB (com o *colormap* adequado).

4. Pré-processamento da imagem: ***padding***.
   4.1. *Encoder*: Crie uma função para fazer padding dos canais RGB. Caso a dimensão da imagem não seja múltipla de 32x32, faça padding da mesma, replicando a última linha e a última coluna em conformidade.
   4.2. *Decoder*: Crie também a função inversa para remover o padding. Certifique-se de que recupera os canais RGB com a dimensão original, visualizando a imagem original.

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.
   5.2. Crie também a função inversa (conversão de YCbCr para RGB). Nota: na conversão inversa, garanta que os valores R, G e B obtidos sejam números inteiros no intervalo {0, 1, …, 255}.
   5.3. *Encoder*:
      5.3.1. Converta os canais RGB para canais YCbCr.
      5.3.2. Visualize cada um dos canais (com o colormap adequado)
   5.4. *Decoder*: Recupere os canais RGB a partir dos canais YcbCr obtidos. Certifique-se de que consegue obter os valores originais de RGB (teste, por exemplo, com o pixel de coordenada [0, 0]).
   5.5. ✍ **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 (*downsampling*) os canais Y, Cb, e Cr, segundo as possibilidades definidas pelo codec JPEG, a qual deve devolver Y_d, Cb_d e Cr_d. Utilize, para o efeito, a função **cv2.resize** (biblioteca *Computer Vision*), testando diferentes métodos de interpolação (e.g., linear, cúbica, etc.).
   6.2. Crie também a função para efectuar a operação inversa, i.e., *upsampling*.
   6.3. *Encoder*: Obtenha e visualize os canais Y_d, Cb_d e Cr_d com downsampling 4:2:0. Apresente as dimensões das matrizes correspondentes.
   6.4. *Decoder*: Reconstrua e visualize os canais Y, Cb e Cr. Compare-os com os originais.
   6.5. ✍ **Apresente e analise o resultado da compressão 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.
      7.1.2. Crie também a função inversa (usando scipy.fftpack.idct).
      Nota: para uma matriz, X, com duas dimensões, deverá fazer:
      X_dct = dct(dct(X, norm="ortho").T, norm="ortho").T
      7.1.3. *Encoder*: Aplique a função desenvolvida em 7.1.1 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** (apta para compressão de gama), de acordo com o seguinte pseudo-código:
      imshow(log(abs(X) + 0.0001))
      7.1.4. *Decoder*: Aplique a função inversa (7.1.2) e certifique-se de que consegue obter os valores originais de Y_d, Cb_d e Cr_d.
   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.
      7.2.2. Crie também a função inversa (IDCT BSxBS).
      7.2.3. *Encoder*: Aplique a função desenvolvida (7.2.1) a Y_d, Cb_d, Cr_d com blocos 8x8 e visualize as imagens obtidas (Y_dct8, Cb_dct8, Cr_dct8).
      7.2.4. *Decoder*: Aplique a função inversa (7.2.2) e certifique-se de que consegue obter os valores originais de Y_d, Cb_d e Cr_d.
   7.3. DCT em **blocos 64x64**. Repita 7.2 para blocos com dimensão 64x64.
   7.4. ✍ **Compare e discuta os resultados obtidos em 7.1, 7.2 e 7.3 em termos de potencial de compressão.**

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

8.4. *Decoder*: Desquantize os coeficientes da DCT, usando os mesmos factores de qualidade. Visualize as imagens obtidas.
8.5. ✍ **Compare os resultados obtidos com os vários factores de qualidade e discuta-os em termos de potencial de compressão.**
8.6. ✍ **Compare os resultados obtidos com os da alínea 7 (DCT) 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.
   9.2. Crie também a função inversa.
   9.3. *Encoder*: Aplique a função 9.1 aos valores da DCT quantizada, obtendo Y_dpcm, Cb_dpcm e Cr_dpcm).
   9.4. *Decoder*: Aplique a função inversa (9.2) e certifique-se de que consegue obter os valores originais de Y_q, Cb_q e Cr_q.
   9.5. ✍ **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. *Encoder*: Codifique as imagens fornecidas com os seguintes parâmetros de qualidade: 10, 25, 50, 75 e 100.
    10.2. *Decoder*: Reconstrua as imagens com base no resultado de 10.1.
    10.3. Crie uma função para cálculo da imagem das diferenças (entre o canal Y da original e da descompactada).
    10.4. Crie uma função para calculo das métricas de distorção MSE, RMSE, SNR, PSNR, max_diff e avg_diff (por comparação da imagem original com a descompactada).
    10.5. ✍ **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, PSNR, max_diff e avg_diff) para cada uma das imagens e factores de qualidade. Tire conclusões.**
    10.6. Volte a analisar a alínea 1, de forma a validar/complementar as conclusões tiradas nesse ponto.