Codificação de Sinais Multimédia

Relatório de Unidade Curricular
Apresentado no Âmbito das Provas de Agregação em Informática

José Manuel Peixoto do Nascimento

Lisboa, 15 de maio de 2023

Conteúdo

1 Introdução

Este relatório foi elaborado no âmbito das provas de agregação na área científica de informática. O documento descreve a motivação, objetivos, programa e métodos de ensino e avaliação da Unidade Curricular (UC) de Codificação de Sinais Multimédia (CSM), inserida do curso de Licenciatura de Engenharia Informática e Multimédia e do curso de Licenciatura em Engenharia Informática e de Computadores do Instituto Superior de Engenharia de Lisboa, Instituto Politécnico de Lisboa.

2 Motivação

No contexto da aquisição, processamento, transmissão e armazenamento da informação, é evidente o rápido crescimento da utilização em larga escala e o rápido desenvolvimento dos sistemas e dos serviços associados à tecnologia digital. Por exemplo, hoje é possível, com um telemóvel, adquirir um sinal de voz, uma imagem ou um vídeo, armazená-lo num servidor e disponibiliza-lo para difusão nas redes sociais. Acresce que, a resolução das câmaras e dos ecrãs é cada vez maior.

Estes sinais com elevada qualidade serão processados e interpretados pelo cérebro humano, pelo que importa manter a sua qualidade. Por outro lado, com o desenvolvimento tecnológico, os sinais ocupam cada vez mais espaço de armazenamento e maior largura de banda na transmissão pondo, em determinados casos, em risco a viabilidade da transmissão em tempo-real. De forma a minimizar o tamanho dos sinais é necessário realizar a sua codificação e compressão, com ou sem perdas.

Estes sinais têm inúmeras aplicações, como por exemplo, nos sistemas de transporte inteligente, em deteção remota, sistemas de vigilância, sistemas de diagnóstico médico, controlo ambiental, entre outros.

Nos dias de hoje, existem diferentes sensores que adquirem sinais que precisam de ser armazenados e/ou transmitidos, sendo para isso necessário fazer a sua codificação. Por exemplo, as câmaras de vídeo, denominadas "analíticas", têm hoje a capacidade de processamento de vídeo incluindo métodos de inteligência artificial, os *scanners* Kinect 3D determinam a profundidade em vários ângulos para compor um modelo tridimensional de um objeto ou de um ambiente, e as câmaras hiperespectrais adquirem várias centenas de imagens em diferentes comprimentos de onda. Para diferentes sensores e dependendo da aplicação será necessário desenvolver novos métodos de

codificação dos sinais adquiridos por estes sensores.

Atualmente a codificação e a compressão de dados é uma área de investigação ativa, em que diferentes técnicas continuam a ser exploradas com o objetivo de se converter os sinais em dados que ocupem o menor espaço possível de armazenamento, a transferência destes seja rápida e eficiente e a descodificação dos dados volte a representar os sinais com fidelidade. Por exemplo, no domínio da codificação de imagens, embora o standard comummente usado seja a norma JPEG, existem neste momento linhas de investigação que se dividem sobretudo em métodos de processamento de imagem e métodos de aprendizagem automática. Os primeiros tipicamente usam modelos de redução de redundância espacial, usando transformações do espaço de sinal, seguido de quantização e codificação entrópica (no caso de codificação com perdas). Nos métodos de aprendizagem automática, destacam-se os métodos de aprendizagem profunda, onde a estrutura da rede com múltiplas camadas é treinada com bases de dados, contendo milhares de imagens, para codificar as imagens minimizando o espaço de armazenamento. As publicações mais recentes sugerem que as normas no futuro deverão aplicar estes métodos dada a sua performance. Estas têm no entanto a desvantagem de necessitarem de uma capacidade de processamento elevada, tempo de aprendizagem elevado e necessidade de bases de dados com elevado número de imagens.

No domínio da codificação de vídeo existem também os métodos tradicionais, que reduzem a redundância temporal e espacial separadamente ou de uma forma híbrida. Nesta vertente são conhecidas as normas H.264 e H.265 e a mais recente *Versatile Video Coding* (VVC), sobre as quais se aplicam novos métodos para melhorar a compressão de vídeo mantendo a performance. A aprendizagem profunda tem recentemente sido aplicada na codificação de vídeo, onde os métodos de predição intra-frame, predição inter-frame e *residual coding* têm vindo a demonstrar desempenho superior aos métodos tradicionais.

Os estudantes de informática devem ter o conhecimento das tecnologias emergentes, assim como as ferramentas teóricas e tecnológicas para abraçar estes novos desafios. No futuro, o desafio tecnológico coloca a engenharia informática como uma atividade fundamental em que os estudantes de hoje, deverão ter formação altamente qualificada para que tenham capacidade para integrar equipas multidisciplinares. A UC de CSM dá a introdução teórica sobre a tecnologia existente e sobre as normas existentes.

3 Objetivos e Competências Adquiridas

Nesta secção apresentam-se os objetivos da UC bem como as competências a adquirir pelos estudantes e a sua integração no curso de Licenciatura em Engenharia Informática e Multimédia.

3.1 Objetivos e Competências

O objetivo fundamental é dar a compreender os fundamentos teóricos que dão suporte à representação, armazenamento e transmissão dos sinais multimédia. Introduzir os vários métodos de compressão e codificação com e sem perdas. Conhecer as principais normas de codificação usadas na indústria multimédia e desenvolver sentido crítico sobre estas.

O processo de ensino/aprendizagem dos tópicos abordados é suportado na realização de um conjunto de pequenos projectos usando a linguagem Python e bibliotecas apropriadas (e.g., Numpy, Scipy, OpenCV, Matplotlib).

É esperado que os estudantes compreendam:

- A necessidade de compressão de sinais;
- O teorema de codificação de fonte;
- Os fundamentos da percepção humana (auditiva e visual) e os modelos de cor em imagem e vídeo;
- Os métodos de codificação sem perdas (códigos entrópicos, códigos baseados em dicionários e codificação aritmética) e os métodos de codificação com perdas (quantificação escalar e vectorial, codificação por transformada e codificação preditiva);
- Os métodos de codificação (com e sem perdas) e os avaliem na representação eficiente de texto, áudio, imagem e vídeo;
- As normas de compressão mais comuns, usadas actualmente na indústria da multimédia e desenvolvam sentido crítico relativamente a estas.

3.2 Integração no Curso

A UC de Codificação de Sinais Multimédia esta inserida no 4º semestre do curso de Licenciatura em Engenharia Informática e Multimédia. Os estudantes antes de frequentarem esta UC, já adquiriram conhecimentos de:

- Álgebra linear na UC de Matemática para a Computação Gráfica, sendo esta essencial para a interpretação geométrica das transformações e para a manipulação de matrizes e vetores;
- Probabilidades na UC de Raciocínio Probabilístico e Simulação. Estes conhecimentos são importantes para o conceito de entropia, necessário à compreensão dos algoritmos de compressão;
- Espaço de sinais, transformada de Fourier e conversão de sinais contínuos para sinais discretos na UC de Processamento Digital de Sinal. Estas noções são essenciais para a compreensão das normas de compressão de imagem e de vídeo;
- Largura de banda, relação sinal-ruído e quantização na UC de Comunicações e Processamento de Sinal. Estas noções são importantes para ter sentido crítico sobre a qualidade do sinal e a taxa de compressão obtida pelas normas de compressão mais comuns;
- Programação e linguagem Python. A algoritmia e capacidade de programação serão essenciais, dado que os projetos a desenvolver para analisar, compreender e avaliar os métodos de compressão serão desenvolvidos em linguagem Python.

A UC de Codificação de Sinais Multimédia é fundamental no enquadramento do curso, uma vez que os seus conteúdos programáticos são alicerces para diversas outras UC no curso. Posteriormente a esta UC, os estudantes são expostos a outras matérias relacionadas com os tópicos nucleares de Codificação de Sinais Multimédia, nomeadamente na UC de Aprendizagem Automática, Processamento de Imagem e Visão, Processamento de Imagem e Biometria e Visão Artificial e Realidade Mista. Sendo estas últimas UC integradas no plano curricular do Mestrado em Engenharia Informática e Multimédia.

A UC é também parte integrante do curso de Licenciatura em Engenharia Informática e de Computadores, sendo oferecida como UC opção no último ano. Os estudantes antes de frequentarem esta UC, têm no seu percurso académico a formação nos conceitos de base para compreender os conceitos do conteúdo programático de CSM.

4 Programa da UC

Nesta secção apresenta-se o resumo do programa da UC, considerando os tópicos fundamentais de cada capitulo. Os estudantes adquirem e consolidam com este programa competências teórico-práticas relacionadas com a codificação de sinais.

4.1 Programa Curricular

1. **Introdução aos sistemas multimédia**

   Introduz ao estudante o conceito de sistemas multimédia, aplicações multimédia (interface com o utilizador, sistemas para autor, sistemas imersivos e de realidade aumentada). São apresentados vários exemplos das variadas profissões e outras utilizações destes sistemas.

   São introduzidos os conceitos da perceção humana e dos diferentes tipos de sinais multimédia: som (áudio e voz), imagem, gráficos, vídeo e texto.

   É apresentada a evolução histórica dos sistemas multimédia, desde a primeira transmissão sem fios, até ao presente. Nesta apresentação, revisita-se os sistemas analógicos de transmissão de sinais de rádio e televisão, para além dos sistemas de armazenamento analógico e digital.

   Introduz-se as noções de codificação de sinais e a necessidade de compressão destes. São abordados os conceitos de redundância (temporal, espacial, estatística), irrelevância, de compressão com e sem perdas e de teoria rácio-distorção.

   Neste capítulo é proposta a elaboração de várias funções usando a linguagem Python. Este contato inicial permite em ambiente laboratorial abordar os temas formulados no contexto teórico e permite também a familiarização dos estudantes com as bibliotecas que irão usar no decorrer das aulas laboratoriais.

2. **Fundamentos de representação de áudio digital**

   Neste capítulo do programa são introduzidos os conceitos físicos do som e da sua propagação (refração, reflexão, difração, atenuação). São abordados os conceitos do sistema auditivo humano e quais as suas limitações, do mascaramento temporal e em frequência. São também abordados os sistemas de aquisição e reprodução de áudio, assim como a digitalização destes sinais (noção de amostragem e aliasing e quantização).

3. **Fundamentos de representação de imagem e vídeo digital**

   Neste capítulo é apresentado o sistema visual humano, é também abordada, a perceção do sistema visual humano e as suas limitações (lei de Weber, retenção temporal, comprimento de onda).

   É introduzido o conceito de representação de cor nos diferentes modelos: RGB, CMY e HSI.

4. **Fundamentos de teoria de informação**

   Neste capítulo, é introduzido o tema da redundância nos sinais, em particular a redundância estatística. São apresentadas as noções básicas da teoria da informação, informação própria, entropia e teorema da codificação de fonte (Shannon's noiseless coding theorem).

5. **Métodos de compressão sem perdas**

   São apresentados neste capítulo vários métodos de codificação sem perdas. Apresenta-se o método *Run length encoding* e as suas vantagens/desvantagens. Com base na teoria da informação são apresentados os algoritmos de comprimento variável (código de Huffman, Código de Shannon-Fano) e a codificação aritmética (decimal e binária). Para a codificação de texto é também abordada a codificação baseada em dicionários (LZ77, LZ78, LZW).

   Neste capítulo é proposta a elaboração e avaliação de um compressor sem perdas usando a linguagem Python e bibliotecas apropriadas.

6. **Compressão de imagem com perdas e codificação por transformada**

   Neste capítulo aborda-se os princípios de compressão com perdas baseada na codificação por transformada. É realizada uma breve revisão da transformada de Fourier para sinais unidimensionais e da transformada de Fourier discreta (DFT). É introduzida a transformada de coseno discreta (DCT) e as suas propriedades de compactação de energia.

   Para aplicar a codificação por transformada em imagens, aborda-se a DCT-2D. Descreve-se a norma JPEG, e os diferentes modos, com exemplo de uma norma de compressão de imagens e apresenta-se o formato JFIF.

   Neste capítulo é proposta a elaboração e avaliação de um compressor com perdas, baseado na norma JPEG, usando a linguagem Python e bibliotecas apropriadas.

7. **Compressão de vídeo, estimação e compensação de movimento**

   Neste capítulo aborda-se os princípios de compressão de vídeo, onde se dá relevância à redundância espacial e temporal. Apresenta-se também a compressão

das componentes de cor (*Chroma-subsampling*). São introduzidos os conceitos da estimação e da compensação de movimento baseada em blocos.

São apresentadas algumas normas de compressão de vídeo, em particular as normas H.26x e MPEGx.

Neste capítulo é proposta a elaboração e avaliação de um compressor de vídeo usando a linguagem Python e bibliotecas apropriadas.

4.2 Bibliografia

- Ze-Nian Li, Mark S. Drew, Jiangchuan Liu, "Fundamentals of Multimedia," Springer, 3ª ed. ISBN: 978-3030621230, 2021.
- Khalid Sayood, "Introduction to Data Compression", Elsevier, 5ª ed., ISBN: 9780128094747, 2017.
- Nuno Ribeiro e José Torres, "Tecnologias de Compressão Multimédia," FCA, ISBN: 978-9727226337, 2009.
- Fernando Pereira, "Comunicações Audiovisuais: Tecnologias, Normas e Aplicações," IST Press, ISBN: 978-9728469818, 2009.

5 Métodos de Ensino e Avaliação

O programa da UC é desenvolvido durante um semestre com duração de quinze semanas. A UC está organizada em aulas teórico-práticas de uma hora e meia e em aulas laboratoriais de três horas, por semana.

5.1 Metodologia de Ensino/aprendizagem

Nas aulas teórico-práticas, aplica-se a metodologia de aprendizagem baseada em projeto (PBL), onde os estudantes são expostos a problemas práticos e são encorajados a encontrar soluções de forma colaborativa com os colegas. Na perspetiva do desenvolvimento e da melhoria contínua, aplica-se também, entre outros, métodos de *Active Learning*, como por exemplo, questionários on-line sobre os tópicos abordados. Os materiais de suporte das aulas estão disponíveis nas plataformas online do ISEL e a bibliografia recomendada encontra-se disponível na biblioteca do ISEL.

Nas aulas laboratoriais, os estudantes consolidam os conhecimentos adquiridos nas aulas teórico-práticas, desenvolvem aplicações que permitam analisar e avaliar os diferentes métodos de codificação para os diferentes sinais, assim como aumentam os seus níveis de desenvolvimento de programação. Estes trabalhos desenvolvem-se em grupos de três estudantes.

O método de ensino/aprendizagem centra-se no aluno, enquanto indivíduo e na sua relação com o professor e com o demais estudantes, promovendo-se o trabalho colaborativo com recurso aos meios tecnológicos disponíveis. Os estudantes são encorajados a procurar soluções noutras fontes de conhecimento, assim como, outras bases de dados de imagens, de vídeos e de sinais onde possam aplicar os métodos por eles desenvolvidos.

5.2 Metodologia de Avaliação

A avaliação é composta por duas componentes,

- uma prova teórica com um peso de 50% na classificação final;
- um conjunto de quatro trabalhos realizados durante o semestre que são apresentados em contexto de aula e sobre o qual é realizado um relatório. Os trabalhos desenvolvidos serão avaliados com base na originalidade, qualidade, eficiência e na justificação para as soluções encontradas, para além da qualidade da escrita do relatório e da apresentação. Esta componente tem um peso de 50% na classificação final.

6 Anexos

6.1 Enunciado de Exame Final

ISEL **Ex 000**

Codificação de Sinais Multimédia

2º Semestre Lectivo 20xx/xx

(xx/xx/20xx)

1. Considere mensagem com 6 símbolos (A; B; E; L; O; R): "BOLABOLAREBOLABOLABOLA".

a) (2 val) Codifique esta mensagem usando o código LZW. Assuma o dicionário inicial [1-"A"; 2-"B"; 3-"E"; 4-"L"; 5-"O"; 6-"R"].

b) (2 val) Calcule a eficiência do código e a taxa de compressão. Explicite todos os pressupostos assumidos.

c) (2 val) Codifique a mensagem usando o algoritmo de Huffman.

d) (2 val) Calcule a entropia associada aos símbolos e compare com o número médio de bits por símbolo usando os dois codificadores.

e) (2 val) Escolha uma das opções:

O número médio de bits por símbolo usado por um codificador sem perdas é:

e1) sempre maior ou igual ao valor da entropia para o conjunto de símbolos da fonte
e2) sempre inferior ao valor da entropia para o conjunto de símbolos da fonte
e3) nenhuma das anteriores

2. Considere a matriz representada.

0	512	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

a) (2 val) Diga justificando o que realiza o código seguinte e o que representa o seu resultado.

```
import cv2
print(cv2.idct(z))
```

b) (2 val) Admitindo que a matriz representa os valores da DCT de um bloco de 8 x 8 após quantificação, a codificação desta segunda a norma JPEG é:

b1) `0011010100001010`
b2) `00111111111000001110000000001010`
b3) `00100001010`
b4) `0011111110110100001010`

3. Considere as norma de compressão de vídeo:.

a) (2 val) O valor da entropia de uma P-frame com compensação de movimento é:

a1) igual ou inferior à entropia da mesma P-frame codificada sem compensação de movimento
a2) igual ou superior à entropia da mesma P-frame codificada sem compensação de movimento
a3) igual à entropia da mesma P-frame codificada sem compensação de movimento

b) (2 val) Represente o diagrama de blocos do modo SNR escalável da norma MPEG2 e apresente as suas vantagens.

c) (2 val) Admita que pretende transmitir um vídeo em UHD 4K (3840x2160). Considere que o fator de compressão é de 100 e 80 para a luminância e crominância respetivamente, que se usa um subsampling de cor 4:2:2, 8 bits por amostra. Determine qual o débito binário assumindo que uma taxa de 30 frames por segundo.

6.2 Enunciado do 1º Trabalho

Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e Multimédia
Codificação de Sinais Multimédia - Trabalho N.º 1

2º Semestre de 20xx/20xx

O ficheiro **Jupiter Notebook** com o relatório e código implementado deve ser submetido no Moodle até xx/xx/20xx.

A biblioteca OpenCV (Open Source Computer Vision Library) tem um conjunto de funções que permite processar imagens em Python. Esta ainda permite trabalhar com vários formatos de ficheiros de imagem.

1. Abra o ficheiro com a imagem "lenac.tif" e apresente a imagem.
   Verifique para que servem os métodos "dtype" e "shape".

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

x_img = cv2.imread("lenac.tif")
cv2.imshow('Original Image', x_img)

print x_img.dtype
print x_img.shape

cv2.waitKey(0)
cv2.destroyAllWindows()
```

2. Grave a imagem que está na variável `x_img` no formato "JPEG" com diferentes qualidades.
   Verifique visualmente a qualidade das imagens assim como o tamanho do ficheiro.
   Produza o código para calcular a taxa de compressão, a SNR e a PSNR.

```
cv2.imwrite('file1.jpg',x_img,(cv2.IMWRITE_JPEG_QUALITY,80))
cv2.imwrite('file2.jpg',x_img,(cv2.IMWRITE_JPEG_QUALITY,10))
```

3. Converta a imagem que está na variável `x_img` para níveis de cinzento, usando o método "cvtColor" e grave o resultado.

Este método aplica a transformação $Y = R * 299/1000 + G * 587/1000 + B * 114/1000$, justifique a utilização desta equação.
Verifique também o tamanho do ficheiro e compare-o com o ficheiro original.

```
x_img_g = cv2.cvtColor(x_img, cv2.COLOR_BGR2GRAY)
cv2.imshow('Gray Image',x_img_g)
cv2.imwrite('file3.bmp',x_img_g)
```

4. Apresente o histograma da imagem (em tons de cinzento) que está na variável `x_img_g`, verifique quantos níveis de cinzento tem a imagem.

```
plt.hist(x_img_g.ravel(),256,[0,256])
```

5. Nos próximos trabalhos será necessário realizar operações com os valores de cada pixel. Para este efeito pode-se transformar a imagem para um array. O código seguinte representa o pixel mais significante da imagem. Apresente oito imagens, cada uma com o valor de cada bit para todos os pixeis.

```
y = x_img_g > 128
cv2.imshow('BW', y*1.0)
```

6. Construa uma função que realize o algoritmo de dithering Floyd Steinberg. Esta função recebe uma matrix (com os pixeis em tons de cinzento) e devolve uma matrix com valores a preto e branco. Este algoritmo aproxima cada pixel da imagem (x) ao valor mais próximo (preto ou branco) e o erro é difundido para os pixeis adjacentes seguindo o método:

$$\begin{bmatrix} . & . & . \\ . & x & 7/16 \\ 3/16 & 5/16 & 1/16 \end{bmatrix}$$

Aplique esta função à imagem em tons de cinzento que está na variável `x_img_g`.

```
y = dither(x_img_g)
```

7. Construa uma função para gravar a matriz obtida na questão anterior (variável `y`) para um ficheiro binário. Verifique o tamanho do ficheiro inicial e do ficheiro final. Calcule a taxa de compressão e meça o SNR e o PSNR.

6.3 Enunciado do 2º Trabalho

Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e Multimédia
Codificação de Sinais Multimédia - Trabalho N.º 2

2º Semestre de 20xx/20xx

O ficheiro **Jupiter Notebook** com o relatório e código implementado deve ser submetido no Moodle até xx/xx/20xx.

Este Trabalho explora os conceitos de compressão de dados sem perdas baseados na teoria da informação.

1. Elabore uma função ("gera_huffman") que gere uma tabela com o código binário para cada símbolo de um dado conjunto, usando o método de Huffman. Esta função deve ter como parâmetros de entrada um conjunto de símbolos e as suas probabilidades (ou em alternativa pode usar o número de ocorrências de cada símbolo, dado pelo seu histograma). Também pode em alternativa gerar não uma tabela mas outra estrutura de dados com os códigos pretendidos.

2. Elabore uma função ("codifica") que dada uma mensagem (sequência de símbolos) e a tabela da ponto anterior, retorne uma sequência de bits com a mensagem codificada.

3. Elabore uma função ("descodifica") que dada uma sequência de bits (mensagem codificada) e a tabela do ponto 1, retorne uma sequência de símbolos (mensagem descodificada).

4. Elabore uma função ("escrever") que dada uma sequência de bits (mensagem codificada) e o nome do ficheiro, escreva a sequência de bits para o ficheiro.

5. Elabore uma função ("ler") que dado o nome do ficheiro, leia uma sequência de bits (mensagem codificada) contida no ficheiro.

6. Teste as funções elaboradas usando para o efeito a imagem fornecida ("lena" em tons de cinzento).

    a) Gere o código usando a função realizada no ponto 1. Meça o tempo que demora a função.

    b) Meça a entropia e o número médio de bits por símbolo. Calcule a eficiência.

    c) Faça a codificação da mensagem contida no ficheiro (usando a função realizada no ponto 2). Meça o tempo que a função demora a fazer a codificação.

d) Grave um ficheiro com a mensagem codificada, usando a função realizada no ponto 4. Veja o tamanho do ficheiro.

e) Leia do ficheiro o conjunto de bits, usando a função realizada no ponto 5.

f) Faça a descodificação da mensagem (usando a função realizada no ponto 3.) Meça o tempo que a função demora a fazer a descodificação.

g) Compare a mensagem descodificada com a original e verifique que são iguais (erro nulo).

7. O relatório deve conter uma descrição breve das funções realizadas e uma tabela com todos os resultados extraídos.

Exemplo para o ficheiro com imagem

```
from time import time
from os import path
import cv2
import matplotlib.pyplot as plt

# Lê a imagem em níveis de cinzento
x = cv2.imread("lena.tiff",cv2.CV_LOAD_IMAGE_GRAYSCALE)

# Converte a imagem (matriz) numa sequência de números (array)
xi = x.ravel()

# Calcula o histogram
h, bins, patches = plt.hist(xi,256,[0,256])

# Gera o código de Huffman
t0 = time()
tabela_codigo = gera_huffman(np.arange(0,256),h)
t1 = time()
print "time:", t1-t0

# Codifica e grava ficheiro
seq_bit0 = codifica(xi,tabela_codigo)
escrever(seq_bit0, filename)
t2 = time()
print "time:", t2-t1
```

```
# Lê ficheiro e descodifica
seq_bit1 = ler(filename)
yi = descodifica(seq_bit1,tabela_codigo)
t3 = time()
print "time:", t3-t2

size_ini = path.getsize("filename original image")
size_end = path.getsize("filename compressed")
print "taxa: ", 1.* size_ini / size_end

plt.show()
cv2.waitKey(0)
plt.close("all")
cv2.destroyAllWindows()
```

6.4 Enunciado do 3º Trabalho

Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e Multimédia
Codificação de Sinais Multimédia - Trabalho N.º 3

2º Semestre de 20xx/20xx

O ficheiro **Jupiter Notebook** com o relatório e código implementado deve ser submetido no Moodle até xx/xx/20xx.

Este trabalho explora os princípios básicos da norma JPEG (modo sequencial) para compressão de imagens com perdas. Neste trabalho apenas serão exploradas as imagens em níveis de cinzento.

1. Construa uma função (codificador) que para cada bloco de 8×8 da imagem original efectue a DCT bidimensional. Use por exemplo a instrução

   ```
   from scipy.fftpack import dct, idct
   bloco_dct = dct(dct(bloco.T, norm='ortho').T ,  norm='ortho')
   ```

   Veja a imagem com o conjunto dos blocos após a transformada.
   Construa uma função (descodificador) que faça a DCT inversa.
   Verifique que a imagem é igual à original.

2. Construa uma função (codificador) que para cada bloco de 8×8 de coeficientes da transformação efectuada faça a divisão pela matriz de quantificação (tabela K1 no anexo da norma) multiplicada por um factor de qualidade q (ver pág. 230 do livro "Tecnologias de Compressão Multimédia").
   Veja a imagem com o conjunto dos blocos após a quantificação.
   Construa uma função (descodificador) que realize a operação inversa da quantificação.
   Junte estas funções às já realizadas e verifique para diferentes factores de qualidade qual a SNR e veja a imagem descodificada.

3. Construa uma função (codificador) que faça a codificação diferencial dos coeficientes DC após a quantificação.
   Construa a função inversa para o descodificador.

4. Construa uma função (codificador) que faça crie um array com a indexação em zig-zag dos coeficientes AC após a quantificação e crie um array com os pares (zero run length, nonzero value).
   Construa a função inversa para o descodificador.

5. Junte estas funções às já realizadas e veja a imagem descodificada.

6. Construa uma função que dados os arrays das alíneas anteriores use as tabelas do código de Huffman (tabela K3 e K5) e grave num ficheiro a sequência de bits correspondente. (não é necessário usar o formato JFIF)

7. Construa uma função que leia o ficheiro gravado e retorne os arrays com os coeficientes AC e DC.

8. Junte estas funções às já realizadas e veja a imagem descodificada.
   Para diferentes factores de qualidade meça a relação sinal-ruído e a taxa de compressão obtida. Represente um gráfico onde se apresente a taxa de compressão em função do SNR.

9. No mesmo gráfico compare o seu compressor de imagem com outros existentes para várias qualidades, por exemplo use:

```
from PIL import Image
x = Image.open("lena.tiff")
x.save("lena_c.jpeg", "JPEG", quality=50)
```

10. O relatório deve conter uma descrição breve das funções realizadas e uma tabela com todos os resultados da SNR, taxa de compressão, tempo de compressão e descompressão.

6.5 Enunciado do 4º Trabalho

Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e Multimédia
Codificação de Sinais Multimédia - Trabalho N.º 4

2º Semestre de 20xx/20xx

O ficheiro **Jupiter Notebook** com o relatório e código implementado deve ser submetido no Moodle até xx/xx/20xx.

Este trabalho explora os princípios básicos da codificação de vídeo. Neste trabalho implementa-se três formas de codificação de vídeo. Para cada um destes deve usar o ficheiro disponível ou outros que queira considerar à sua escolha. Para cada codificador deve medir (frame a frame):

1. a taxa de compressão;
2. a relação sinal-ruído (relação entre a frame original no emissor e a frame final no receptor;
3. a entropia da frame a transmitir;
4. a energia da frame a transmitir;
5. o tempo de compressão e descompressão.

No final deve construir gráficos ou tabelas que ilustrem estas medidas em função da frame.

As formas de codificação são:

1. Considerar que cada frame é uma intra-frame (I). Pode usar o codificador do trabalho anterior ou usar o codificador já disponível:

```
from PIL import Image
x = Image.open("bola_0x.tiff")
x.save("bola_0x.jpeg", "JPEG", quality=50)
```

2. Considerar que cada frame à excepção da primeira são inter-frames (P), ou seja, todos os macroblocos são do tipo (P). Neste codificador deve criar as P-frames, que são a diferença entre a frame a codificar e a I-frame, sem compensação de movimento. Visualize a P-frame (ou seja a imagem a transmitir).

3. Considerar que cada frame à excepção da primeira são inter-frames (P). Neste codificador deve implementar a predição da frame a codificar com base na I-frame fazendo a compensação de movimento. A frame a transmitir é a diferença entre entre a frame codificar e a sua predição. Sugere-se a construção de três funções:

   (a) uma função para fazer a medição do erro absoluto médio entre dois blocos (tamanho 16×16);

   (b) uma função que faça uma pesquisa (pode escolher a full-search ou outra) do bloco da frame a codificar numa janela de pesquisa (-15 a +15) da I-frame;

   (c) uma função que percorra os blocos da frame a codificar e construa a frame predita;

   Visualizar a frame predita, e a frame diferença, bem como os vectores de movimento (use a função pylab.quiver para o efeito).