



FACULDADE DE
CIÉNCIAS E TECNOLOGIA
UNIVERSIDADE DE
COIMBRA

dei25

departamento
de engenharia informática
1995 - 2020

Relatório

Trabalho 5 - Codificação JPEG

Unidade Curricular:
Processamento Audiovisual (PA)

Licenciatura em Engenharia e Ciência de Dados

Realizado por:

Diogo Beltran Dória, 2020246139
Mariana Lopes Paulino, 2020190448

Ano Letivo 2022/2023

Introdução

O objetivo deste trabalho é executar várias experiências envolvendo o codificador JPEG, usando os seus vários modos de codificação.

Experiências

Experiência 1 - Comparação entre codificação com perda e sem perda

1 - Comprimimos a imagem CrowdRun em modo JPEG lossless (sem perda) usando o comando dado:

```
jpeg -l -q 100 -c CrowdRun.ppm CrowdRunLS.jpg
```

Em que as opções `-l -q 100` especificam codificação com qualidade máxima (lossless) com uso de uma DCT 1D reversível.

De seguida é pedido para calcular o valor aproximado da taxa de compressão TO/TC, em que TO corresponde ao tamanho (nº de bytes) do ficheiro da imagem original e TC ao tamanho do ficheiro da imagem comprimida. O tamanho da imagem CrowdRun.ppm é de 24.9MB. E o tamanho da imagem comprimida CrowdRunLS.jpg é de 18.8MB. Logo $TO/TC = 1.32446809$



Figura 1. CrowdRunLS.jpg

2 - Comprimimos de novo a imagem original, mas agora em modo lossy (com perda), com factor de qualidade 20 (opção `-q 20`), usando o comando:

```
jpeg -q 20 CrowdRun.ppm CrowdRunQ20.jpg
```

Obtendo um tamanho da imagem comprimida de 584 KB. Logo $TO/TC = 42.6369863$.



Figura 2. CrowdRunQ20.jpg

Nestas duas imagens o utilizador consegue verificar algumas diferenças em ambas as imagens quando as comparamos, uma vez que a qualidade das mesmas difere nomeadamente nas áreas mais claras da fotografia, ficando essas mais pixelizadas. Nas outras áreas da imagem também é notável a pixelização dos vários elementos da mesma.

Na imagem original, conseguimos notar uma qualidade melhor e uma quantidade de píxeis inferior ao que acontece na imagem, onde a operação realizada é a perda de qualidade. No entanto, as imagens ainda são bastante parecidas, apenas difere nesse pormenor de qualidade em áreas mais claras da imagem.

3 - Fizemos várias compressões com qualidades $q = 25, 50, 75, 100$. Obtendo as seguintes taxas de compressão para cada:

$$\begin{aligned} q &= 25, \text{ TC} = 675 \text{ KB}, \text{ TO/TC} = 36.8888889 \\ q &= 50, \text{ TC} = 1.1 \text{ MB}, \text{ TO/TC} = 22.6363636 \\ q &= 75, \text{ TC} = 1.9 \text{ MB}, \text{ TO/TC} = 13.1052632 \\ q &= 100, \text{ TC} = 16.7 \text{ MB}, \text{ TO/TC} = 1.49101796 \end{aligned}$$

Concluímos que quanto menor for a qualidade maior é a taxa de compressão



Figura 3. CrowdRunQ25.jpeg



Figura 4. CrowdRunQ50.jpeg



Figura 5. CrowdRunQ75.jpeg



Figura 6. CrowdRunQ100.jpeg

Experiência 2 – Efeito das tabelas de quantização

4 - O codificador JPEG fornecido suporta várias tabelas de quantização dos coeficientes da DCT. Entre elas está uma tabela optimizada tendo em conta as características dos sistema visual humano (opção `-qt 4`). Em que “`-qt4`” especifica o uso dessa tabela de quantização otimizada, usando o comando:

```
jpeg -q 20 -qt 4 CrowdRun.ppm CrowdRunQ20QT4.jpg
```

Obtivemos:



Figura 7. CrowdRunQ20QT4.jpg

Em comparação, a imagem CrowdRunQ20QT4.jpg tem mais qualidade que a imagem CrowdRunQ20 uma vez que a quantização utilizada foi otimizada, acabando a fazer com que a imagem não perca tanta qualidade mantendo também as cores originais. A imagem

CrowdRunQ20QT4 é mais parecida à original, mantendo as suas características e a sua qualidade não sofre alterações de maior que sejam muito notáveis à distância.

Quanto ao tamanho dos ficheiros a CrowdRunQ20 tem um tamanho de 584 KB sendo este tamanho inferior ao da CrowdRunQ20QT4 cujo tamanho é de 659 KB, concluindo assim que os ficheiros não apresentam uma grande variação no tamanho pelo que podemos concluir que as variações nas imagens também não são muito grandes.

Em suma, concluímos que o ficheiro com maior tamanho tem um pouco mais qualidade que o outro (CrowdRunQ20) o que também conseguimos provar com as comparações feitas através da observação de ambas as imagens.

Experiência 3 – Codificação progressiva

5 - O codificador JPEG fornecido suporta codificação progressiva. Codificando a mesma imagem original usando o comando:

```
jpeg -v -q 50 CrowdRun.ppm CrowdRunProg.jpg
```



Figura 8. CrowdRunProg.jpg Tamanho: 1.1MB

De seguida, usando um editor binário ou um comando Unix como:

```
head -c n CrowdRunProg.jpg > CrowdRunTruncn.jpg
```

truncamos o ficheiro conservando só os **n** primeiros bytes do mesmo.

Criamos dez ficheiros truncados derivados de CrowdRunProg.jpg, para **n** igual a 20.000, 70.000, 90.000, 120.000, 160.000, 300.000, 350.000, 400.000, 450.000 e 500.000



Figura 9. CrowdRunTrunc20000.jpeg Tamanho: 20KB



Figura 10. CrowdRunTrunc70000.jpeg Tamanho: 70KB



Figura 11. CrowdRunTrunc90000.jpeg Tamanho: 90KB



Figura 12. CrowdRunTrunc120000.jpeg Tamanho: 120KB



Figura 13. CrowdRunTrunc160000.jpeg Tamanho: 160KB



Figura 14. CrowdRunTrunc300000.jpeg Tamanho: 300KB



Figura 15. CrowdRunTrunc350000.jpeg Tamanho: 350KB



Figura 16. CrowdRunTrunc400000.jpeg Tamanho: 400KB



Figura 17. CrowdRunTrunc450000.jpeg Tamanho: 450KB



Figura 18. CrowdRunTrunc500000.jpeg Tamanho: 500KB

Com a codificação progressiva notamos que existe um crescimento da qualidade gradual nas imagens e também um crescimento gradual do tamanho do ficheiro. É de notar que à medida que as imagens vão ficando maiores vão aparecendo mais detalhes da imagem e a imagem vai ficando completa. Após as imagens ficarem completas, com o aumento da truncção dos n bits das imagens vamos também garantido um nível de detalhe superior a cada experiência que vamos fazendo.

A codificação progressiva é um método bastante bom para a codificação de imagens uma vez que em vez de efetuar um único scan à imagem e aos seus detalhes e, ao aumentar os bits truncados, vamos aumentando os detalhes das imagens e experiências realizadas, conseguindo se aumentarmos bastante o número de bits truncados iremos obter uma imagem

bastante similar à original, não tendo um nível de detalhe tão elevado mas onde dá para perceber todos os seus detalhes na mesma.

Experiência 4 – Codificação Entrópica

6 - O codificador JPEG fornecido suporta codificação Huffman, codificação aritmética e ainda Huffman com optimização dos códigos de Huffman.

a. Codificamos a mesma imagem original (neste caso foram usadas as tabelas de Huffman pré-definidas) usando o comando:

```
jpeg -q 20 CrowdRun.ppm CrowdRunQ20.jpg
```



Figura 19. CrowdRunQ20.jpg Tamanho: 584 KB

b. Codificamos a mesma imagem original (neste caso foi usada **codificação aritmética** (opção `-ar`) usando o comando:

```
jpeg -q 20 -ar CrowdRun.ppm CrowdRunQ20AR.jpg
```



Figura 20. CrowdRunQ20AR.jpg Tamanho: 584 KB

c. Por último codificamos a mesma imagem original. Neste caso usamos **tabelas de Huffman optimizadas** (opção -h). Com o comando:

```
jpeg -q 20 -h CrowdRun.ppm CrowdRunQ20HOPT.jpg
```



Figura 21. CrowdRunQ20HOPT.jpg Tamanho: 505 KB

Para a) ./jpeg -q 20 CrowdRun.ppm CrowdRunQ20.jpg 0.48s user 0.03s system 98% cpu 0.519 total

Para b) ./jpeg -q 20 -ar CrowdRun.ppm CrowdRunQ20AR.jpg 0.01s user 0.01s system 84% cpu 0.019 total

Para c) ./jpeg -q 20 -h CrowdRun.ppm CrowdRunQ20HOPT.jpg 0.56s user 0.03s system 99% cpu 0.595 total

Das três codificações utilizadas, a mais rápida foi a codificação aritmética que gerou um ficheiro de tamanho igual ao da codificação que utilizou as tabelas de Huffman. A última

codificação, que utilizou tabelas de Huffman otimizadas, gerou um ficheiro menor que ambas as outras codificações. Em termos de eficiência de codificação entrópica dos vários ficheiros, a codificação aritmética foi a mais eficiente, resultando desta um menor tempo de execução e um ficheiro com igual tamanho aos resultantes das tabelas de Huffman. Relativamente aos tamanhos obtidos, a codificação partindo das tabelas de Huffman otimizadas foi o que gerou um ficheiro menor de 505 KB de tamanho em 0.595 segundos no total. Ambas as outras codificações geraram um ficheiro de 584 KB. Na codificação partindo das tabelas de Huffman este tamanho de ficheiro foi obtido em 0.519 segundos no total, e a optimização aritmética obteve um ficheiro de 584 KB em 0.019 segundos no total, sendo esta a mais eficiente em termos da relação rapidez/qualidade.