CODECs não destrutivos

Técnicas de compressão aplicadas a imagens monocromáticas

João Afonso Vieira de Sousa, 2019224599, uc2019224599@student.uc.pt

José Domingos da Silva, 2018296125, uc2018296125@student.uc.pt

Sancho Amaral Simões, 2019217590, uc2019217590@student.uc.pt

Tiago Filipe Santa Ventura, 2019243695, uc2019243695@student.uc.pt

Universidade de Coimbra, Licenciatura em Engenharia Informática
Segundo ano, primeiro semestre, Teoria da Informação

***Abstrato*—Este relatório pretende abordar métodos de compressão destrutivos e não destrutivos, focando-se sobretudo nestes últimos. Relativamente à parte mais teórica do mesmo, proceder-se-á à enumeração de alguns filtros, transformadas e algoritmos mais utilizados no âmbito da compressão de dados. Estes serão então explanados, analisados e comparados a fim de se deduzirem as aplicações mais adequadas para cada um destes. Quanto à parte experimental deste relatório, diversas soluções para a compressão de imagens monocromáticas (em especial, do *dataset* proposto) serão apresentadas, devidamente justificadas e comparadas com a solução *baseline* fornecida, o *PNG*.**

Palavras-chave—destrutiva, não destrutiva, compressão, filtros, transformadas, monocromáticas, dataset, PNG

I. INTRODUÇÃO

Na atualidade, com a quantidade incomensurável de informação processada e enviada diariamente, tornou-se cada vez mais impreterível a engenharia de métodos para a flexibilização e utilização menos dispendiosa e menos exigente, em termos de infraestrutura, do fluxo dessa mesma informação. A compressão, mais que uma mais-valia, transformou-se numa necessidade. A menos que se trabalhe com supercomputadores no quotidiano, unidades de armazenamento de gigantescas dimensões e exageradas larguras de banda, nunca seria possível acompanhar o ritmo necessário ao funcionamento do próprio mundo, tão dependente da rápida partilha de conhecimentos e saberes entre indivíduos separados por grandes distâncias. Tomemos como exemplo a corrente pandemia do COVID-19: a elaboração de uma vacina provou ser um trabalho hercúleo, sendo extremamente dependente de estudos e ensaios clínicos sobre as mais variadas receitas para a vacinação. Empresas, farmácias e centros clínicos de todo o mundo, mesmo enquanto este relatório está a ser escrito, trabalham sem parar, confrontando as suas descobertas com as dos seus pares do outro lado do globo. Este é um claro exemplo da necessidade de transmissão de informação em tempo e espaço reduzido. Para tal objetivo ser concretizado é, portanto, necessário recorrer a técnicas de compressão de dados, que se podem dividir em dois tipos: *lossless* (não destrutiva) e *lossy* (destrutiva).

II. COMPRESSÃO *LOSSY* VS COMPRESSÃO *LOSSLESS*

A. Compressão lossy

Em muitos casos, é necessário comprimir um determinado conjunto de dados de modo a que, quando descomprimido, a informação seja idêntica à inicial, isto é, durante o processo de compressão, a informação não é adulterada – compressão *lossless* (não destrutiva ou entrópica). Atente-se neste exemplo: a compressão do texto não deve alterar o seu conteúdo, uma vez que isto implicaria a modificação de carateres e, portanto, poderia fazer com que certos segmentos textuais deixassem de fazer sentido e/ou adquirissem sentidos diferentes. Entre as várias técnicas de compressão *lossless* encontram-se os algoritmos de *Lempel-Ziv* (*LZ77*, *LZ78*, *LZMA*, *LZW*, …), para propósito geral, *Dolby TrueHD*, para áudio, e *PNG*, para imagens.

B. Compressão lossy

Os sentidos do corpo humano, tais como a visão e a audição, apresentam limitações, isto é, existem limiares (frequências), a partir dos quais os sinais (eletromagnéticos e sonoros) se tornam indetetáveis. A compressão *lossy* (destrutiva) tira partido deste facto: trechos de um conjunto de dados considerados impercetíveis ou descartáveis para o ser humano são removidos em troca de um ganho substancial de espaço. Alguns exemplos de algoritmos de compressão *lossy* são: *JPEG*, para imagens e *Dirac*, para vídeo.

III. TÉCNICAS DE COMPRESSÃO *LOSSLESS*

O processo utilizado por um típico *codec lossless* pode ser resumido através do seguinte diagrama:

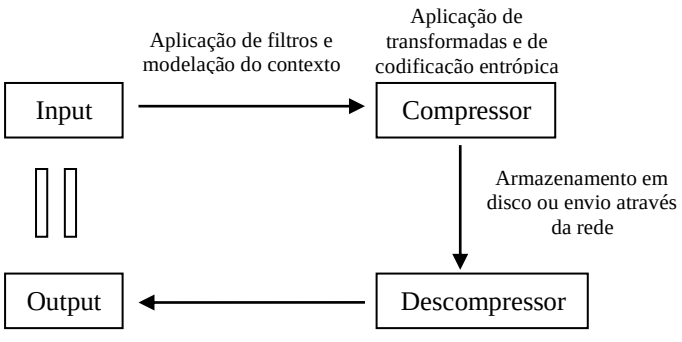

Fig.1: Diagrama simplificado da estrutura de um *codec*

Os filtros e transformadas são transmutações reversíveis aplicadas aos dados que possuem o propósito de tornar o *input* mais comprimível. É de salientar que, no entanto, as últimas apresentam uma vertente algorítmica mais complexa. Alguns exemplos destes mecanismos serão apresentados adiante.

A modelação do contexto é uma etapa na qual ocorre a definição de um modelo probabilístico que utiliza o contexto onde os símbolos ocorrem (através dos símbolos precedentes) para determinar o número de bits utilizados para os codificar.

O compressor e descompressor possuem o papel de, como o próprio nome indica, comprimir e descomprimir os dados, de modo a que o *Input* seja idêntico ao *Output*, uma vez que se trata de compressão não destrutiva.

A. Filtros

Os filtros, também designados por métodos de predição, quando aplicados a um conjunto de dados, reduzem a entropia deste, isto é, potenciam a sua redundância estatística.

De seguida, apresentam-se alguns destas transformações comummente utilizados no algoritmo de compressão *PNG - delta filters*. [1].

1) Sub

Cada *byte* é substituído pela sua diferença com o anterior/posterior.

2) Up

Cada *byte* é substituído pela sua diferença com o de cima.

3) Average

Cada *byte* é substituído pela sua diferença com a média entre o símbolo anterior e posterior, truncando-se qualquer parte decimal resultante.

4) Paeth (Alan W. Paeth, 1956)

Cada byte é substituído pela sua diferença com o preditor de *Paeth* dos bytes anterior, acima e superior esquerdo. (P_{Paeth} = *left + above – upperleft*)

Segue abaixo um exemplo claro de como os filtros, neste caso o filtro *sub*, reduzem substancialmente a dispersão de uma fonte.

Sub

X = [1, 2, 3, 4, 5, 4, 3, 4, 3, 2, 1]

X´= [1, 1, 1, 1, 1, -1, -1, 1, -1, -1, -1]

$H(X) \approx 2.231,\ H(X') \approx 0.994$

B. Transformadas

As transformadas são aplicadas a um conjunto de dados de modo a que os algoritmos de compressão possam depois atuar de forma mais eficiente. Algumas permitem reduzir a dispersão dos dados e outras apenas agrupam símbolos de contexto semelhante e, como a entropia não tem em conta a ordem, permanece inalterada. De seguida, são apresentadas algumas das transformadas mais utilizadas no âmbito da compressão de dados.

1) Burrows-Wheeler Transform (BWT)

Esta transformada permite aumentar a redundância espacial de um dado conjunto de dados, agrupando símbolos de contexto idêntico. [2]

Aplicação da *BWT*:

a) Gerar a matriz de rotações da *string* a codificar.

b) Ordenar lexicograficamente as linhas da matriz obtida.

A *string* codificada é a última coluna. É necessário guardar o índice da linha da matriz onde a sequência original aparece juntamente com a *string* codificada para possibilitar uma sua posterior reconstrução.

Inversão da *BWT*:

Reconstrução da matriz de rotações da *string* original:

i) Começar com a *string* codificada na última coluna

ii) Ordenar lexicograficamente as linhas da matriz.

iii) Prefixar a(s) coluna(s) atuais com a coluna correspondente à *string* codificada.

iv) Voltar ao passo ii) até que se complete a matriz de rotações (ordenada).

A *string* original (descodificada) é a que consta na linha de índice igual ao guardado na fase de codificação.

Nota: Existem algoritmos mais eficientes para a aplicação da BWT, nomeadamente através de um *array* de sufixos da *string* a codificar.

2) Move to front (MTF)

A ideia principal por detrás deste algoritmo consiste em ter uma lista com os carateres do alfabeto da fonte a codificar. Habitualmente, esta é procedida da aplicação de uma transformada *BWT* ou de um *RLE* (*Run Length Encoding*). [3]

Aplicação da *MTF*:

i) Ler o símbolo atual da *string* a codificar. O código que lhe é atribuído corresponde ao índice da posição onde aparece na lista de símbolos.

ii) Mover o símbolo em questão para a frente da lista.

iii) Voltar ao passo i) até que se complete o processamento de toda a *string*.

3) Transformada discreta de cosseno (DCT – Discrete Cosine Transform - Nasir Ahmed, 1972)

A *DCT* permite definir um conjunto discreto de dados (no caso das imagens, utilizando-se usualmente blocos de 8x8 pixeis) através da sobreposição de funções de cosseno, definida numa sequência finita de pontos. É de notar que cada função destas possui uma componente em x e em y.

Dada a finitude do conjunto em questão, é possível enumerar todas as possibilidades de funções de cosseno. Qualquer bloco depois pode ser gerado através de uma combinação linear dessas mesmas funções, cujos coeficientes são integrados na codificação.

É importante salientar que a DCT foi inicialmente desenvolvida para ser aplicada em algoritmos de compressão *lossy*, uma vez que permite remover detalhes impercetíveis pelo olho humano (correspondentes às frequências mais elevadas). No entanto, é também utilizada no âmbito da compressão não destrutiva. [4][8]

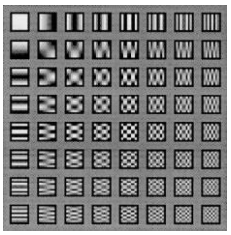

Fig.2: Tabela com todas as possibilidades das funções de cosseno para blocos de 8x8 píxeis
Fonte: Adaptado de [4]

C. Algoritmos de compressão

Com a evolução das necessidades, os *codecs* (não destrutivos) foram-se especializando para tratar determinados tipos de dados. No entanto, atualmente, a maioria apresenta visíveis similitudes, nomeadamente, no que diz respeito aos algoritmos base utilizados. Dentro destes algoritmos base, estão: codificação de *Huffman*, codificação aritmética, codificação por dicionários (família *Lempel-Ziv*). Ou seja, qualquer técnica de compressão *lossless* moderna, contém, algures no seu núcleo, pelo menos um dos algoritmos referidos (ou variantes destes) a correr. Alguns destes mesmos algoritmos serão a seguir brevemente descritos.

1) Algoritmos base de compressão lossless

a) Run Length Encoding (RLE)

Este algoritmo tira partido da redundância espacial de um dado conjunto de dados. Corridas de um dado símbolo *a*, de tamanho $n > 3$, são substituídas pelo *token* (⌙, *a*, *n*), onde ⌙ é um carater de escape predefinido. [6]

b) Codificação de Huffman (Huffman Encoding - David A. Huffman, 1952

A codificação de *Huffman* parte do princípio que, aos símbolos com maior probabilidade de ocorrência, deve ser atribuído um código de menor comprimento. É construída uma árvore binária cujas folhas correspondem aos símbolos a codificar e os nós intermédios são símbolos compostos, cujo valor de probabilidade é a soma das probabilidades dos seus ascendentes. Esta técnica permite obter códigos de prefixo e, portanto, unicamente descodificáveis. [7]

c) LZ-77 (Abraham Lempel, Jacob Ziv, 1977)

O *LZ-77* é um código de dicionário que permite tirar partido da repetição de sequências de símbolos ao longo de um conjunto de dados e, em oposição ao que acontece nos códigos de *Huffman*, não necessita de conhecimento à *priori* da fonte. São utilizadas duas estruturas: o *search buffer* (*SB*), correspondente à janela com os últimos símbolos N_S processados e o *look-ahead buffer* (*LB*), que contém os próximos N_L símbolos por processar.

A ideia central deste algoritmo é a de procurar no *SB* o maior padrão que ocorre no *LB*. Caso esse padrão se encontre, é transmitido o *token* (*offset*, *l*, *next*), onde o *offset* é a distância relativa do padrão encontrado à posição atual do *SB*, *l* o seu comprimento e *next* o código do próximo símbolo.

Apesar de o *LZ-77* não utilizar qualquer conhecimento estatístico sobre a fonte, o seu desempenho aproxima-se muito do de codificadores entrópicos. Uma vez que o tamanho do *SB* é limitado, não é possível encontrar padrões repetidos de grandes dimensões. Portanto, como alternativas mais eficientes e também mais usadas na atualidade, existem o *LZ-78* e o *LZW*. [9]

2) Algoritmos modernos de compressão lossless

a) Bzip2 (Julian Seward, 1996)

O *Bzip2* é um algoritmo de compressão capaz de comprimir qualquer tipo de ficheiro individualmente, não podendo atuar sobre arquivos. Esta técnica de compressão *lossless* apresenta múltiplas camadas de compressão. [6]

Etapas de compressão do *Bzip2*:

i) *RLE* dos dados iniciais.
ii) Transformada *BWT* em blocos de até 900 KB.
iii) Transformada *MTF*.
iv) *RLE* do resultado da transformada *MTF*.
v) Codificação de *Huffman*.
vi) Seleção entre várias tabelas de *Huffman*.
vii) Codificação unária da seleção da tabela de *Huffman* (corrida de 1´s seguida de um 0).
viii) Aplicação do filtro *sub* nos tamanhos dos símbolos de cada tabela de *Huffman* selecionada.

b) Deflate (Phil Katz)

O algoritmo de compressão *deflate* foi inicialmente desenvolvido para ser utilizado em ferramentas de compactação de arquivos como *PKZIP* e *GZIP* (*.zip*). [7]

Etapas de compressão do *Deflate*:

i) Divisão dos dados em blocos.
ii) Aplicação do *LZ-77* em cada bloco.
iii) Codificação de *Huffman* com geração de duas árvores: uma para codificar os símbolos e comprimentos das cadeias encontradas pelo *LZ-77* e outra para codificar os *offsets* contidos em cada *token* formado pelo mesmo.

c) Lossless JPEG (1993)

O *Lossless JPEG* foi uma nova funcionalidade adicionada ao *standard JPEG* para permitir a compressão não destrutiva de imagens. [5]

Etapas de compressão do *Lossless JPEG*:

i) Descorrelação dos dados através da aplicação de filtros (métodos de predição).
ii) Aplicação de um codificador entrópico nos resíduos resultantes (códigos de *Huffman* ou códigos Aritméticos, por exemplo).

d) O algoritmo de compressão utilizado no PNG (Portable Network Graphics - 1996)

O *PNG* é o formato de dados para imagens mais utilizado na atualidade, não só pela sua elevada versatilidade, mas também porque dispõe de até 2^{24} cores (*RGB*) e 2^8 graus de transparência (canal *alpha*). Além do mais, a compressão de imagens é efetuada sem qualquer perda de informação, alcançando taxas de compressão bastante elevadas quando comparadas com as de algoritmos de propósito mais geral, como o *Bzip2* ou o *Gzip*.

O *PNG* será abordado com mais detalhe mais à frente neste relatório, uma vez que constitui a solução *baseline* para a compressão do *dataset* de imagens monocromáticas fornecido. [1]

Etapas de compressão do *PNG*:

i) Aplicação de preditores nos dados.
ii) Aplicação do algoritmo *deflate* nos resíduos resultantes do passo anterior.

3) Aplicações dos algoritmos de compressão lossless

Após a enumeração de alguns algoritmos de compressão não destrutiva e sua breve descrição, há que destacar que a existência da diversidade destas técnicas permite satisfazer a variedade de tipos de dados que existem e as particularidades que cada um destes apresenta. No caso das imagens, será mais adequado utilizar algoritmos como o implementado no formato *PNG* ou o *Lossless JPEG* em detrimento de métodos de compressão de propósito mais geral como o *Bzip2*, enquanto que nos ficheiros de texto será mais eficiente utilizar códigos de *Huffman* ao invés de *RLE*.

Dentro do domínio das imagens, podemos ter também especificidades que necessitam de algoritmos mais *well suited*, na medida em que apresentam um melhor desempenho no que toca à compressão de imagens do subdomínio associado. Por exemplo, falando-se do *dataset* que foi proposto (conjunto de imagens em tons de cinzento), que é um subdomínio dentro do domínio das imagens, é possível, certamente, encontrar uma técnica de compressão não destrutiva mais eficiente do que a utilizada no *PNG*, ao serem estudadas meticulosamente as particularidades das imagens monocromática. É exatamente este trabalho de investigação que foi desenvolvido e será explanado adiante, neste relatório.

4) Critérios de escolha de um algoritmo de compressão

Habitualmente, a escolha de um algoritmo de compressão para uma determinada finalidade, tem em conta os seguintes critérios: [8]

- Eficiência de compressão (Taxa de compressão).
- Velocidade de compressão/descompressão.
- Complexidade de implementação.
- Robustez.
- Escalabilidade.

Relativamente à parte experimental deste relatório, será dada preferência a soluções que priorizem a eficiência de compressão do *dataset* fornecido.

REFERÊNCIAS

[1] G. Roelofs, "PNG: The Definitive Guide", Cap. 9, 1999.

[2] A. Martubs, F. Laranjeira, J. Cunga, L. Silva, "Transformada de Burrows-Wheeler", Universidade Fernando Pessoa, http://multimedia.ufp.pt/codecs/compressao-sem-perdas/supressao-de-sequencias-repetitivas/metodo-de-burrows-wheeler/, acedido em 25/11/2020.

[3] A. Kaur, "Move To Front Data Transform Algorithm", GeeksForGeeks, https://www.geeksforgeeks.org/move-front-data-transform-algorithm/, acedido em 25/11/2020

[4] D. Marshal, "The Discrete Cosine Transform (DCT)", 10/04/2001, https://users.cs.cf.ac.uk/Dave.Marshall/Multimedia/node231.html, acedido em 26/11/2020.

[5] "Lossless JPEG", Wikipedia, https://en.wikipedia.org/wiki/Lossless_JPEG, acedido em 25/11/2020.

[6] S. Ikmulwar, D. Kapgate, "A Review on: Lossless Image Compression Techniques and Algorithms", outubro 2014.

[7] A. Gupta, A. Bansal, V. Khanduja, "Modern Lossless Compression Techniques: Review, Comparison and Analysis", Dept. of Information Technology of Delhi

[8] L. J. Karam, "Lossless Image Compression", Arizona State University, Cap. 16, pp 385-419.

[9] P. Carvalho, "Teoria da Informação", Faculdade de Ciências e Tecnlogia da Universidade de Coimbra, Cap. 2, 2020/2021