Estado da Arte: Principais Técnicas de Compressão sem Perdas para Texto e Imagens

Dário Félix
Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia da Universidade de Coimbra
Coimbra, Portugal
dario@student.dei.uc.pt

Jonas Rua
Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia da Universidade de Coimbra
Coimbra, Portugal
rua@student.dei.uc.pt

***Resumo* — Neste trabalho abordamos codecs e as principais técnicas utilizadas atualmente na compressão não destrutiva para texto e imagens. Para tal, introduzimos com conceitos teóricos sobre a compressão, mas também específicos ás imagens e textos. Mais tarde, enumeramos diversas técnicas com descrições breves e as suas aplicações praticas (codecs).**

Palavras-chave — compressão, compressão sem perda, redundância, codificação, entropia, compressão de texto, compressão de imagens, dependência estatística, codecs

I. INTRODUÇÃO

A compressão de dados refere-se ao processo de reduzir uma quantidade de dados necessários para representar uma dada quantidade de informação. Deduz-se, então, que dados não têm o mesmo significado que informação – várias quantidades de dados podem ser usados para representar a mesma quantidade de informação, uma vez que essa representação pode conter informação irrelevante ou repetida, ou seja, dados redundantes (Fig.1). Compressão sem perdas refere-se a métodos de compressão de dados aplicados por algoritmos em que a informação obtida após a descompressão é idêntica à informação original, em oposição à compressão com perda de dados. [1] [2] A compressão é importante devido à crescente quantidade de informação que é criada, armazenada e transmitida. Para escolher a melhor técnica de compressão utilizam-se indicadores tais como taxa de compressão, velocidades de compressão e de descompressão.

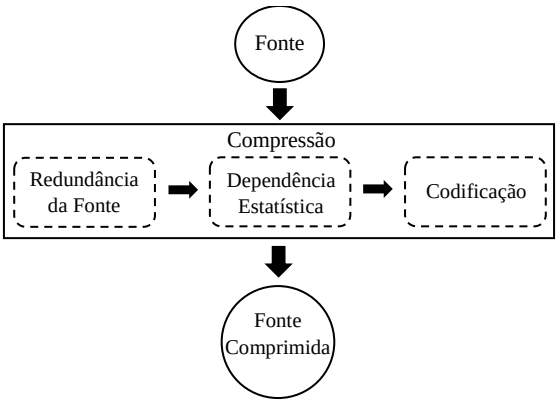

Fig. 1. Técnicas de compressão englobam várias e diferentes fases algorítmicas.

II. TÉCNICAS – CONSIDERAÇÕES SUCINTAS

A. Imagens

Os algoritmos de compressão são desenvolvidos para tirar vantagem da redundância das imagens [1]:

1) Redundância na codificação

Ocorre quando os dados usados para representar a imagem não são utilizados da maneira mais otimizada.

2) Redundância interpixel (espacial)

Ocorre porque pixéis adjacentes tendem a ser altamente correlacionados, uma vez que, por exemplo, na maioria das imagens, o nível do brilho não muda drasticamente de um pixel para outro, mas muda gradualmente.

3) Redundância "psicovisual" (irrelevância)

Há informação que é mais importante para a visão humana que outros tipos de informação. Usado essencialmente na compressão com perdas.

Exemplos da aplicação de compressão sem perdas em imagens são nomeadamente em imagens médicas e legais. [3]

B. Texto

Há muitos dados que contêm informação repetida (e portanto, desnecessária). Por exemplo, a palavra AAAAAA pode ser convertida em 6A, reduzindo o espaço necessário de 6 bytes para 2 bytes.

NOTA - as técnicas de compressão que se seguem são comuns a imagens e texto, bem como a qualquer tipo de dados. Técnicas específicas de imagens são precedidas de *

C. Técnicas de Compressão

1) RLE – Run Length Encoding

Os dados são lidos e os símbolos repetitivos são substituídos por um caracter especial seguido de um símbolo (chamado 'run') e a contagem binária. Como cada repetição requer três caracteres para a representação, uma repetição com menos 4 caracteres não é considerado. [5]

2) Huffman Encoding

A codificação de Huffman é baseada na frequência dos símbolos (entropia) e, portanto, os símbolos com mais frequência obtêm o código binário mais curto, enquanto os símbolos menos frequentes obtêm o mais longo. [5]

3) LZW Encoding – Limpel-Ziv-Welch

A codificação de LZW é baseado na quantidade de multiplicidade de sequências de bits no pixel a ser codificado. Esta é uma abordagem de compressão sem erros que se concentra na eliminação da redundância espacial . Atribui palavras de código de comprimento fixo para sequências de comprimento variável de símbolos da fonte. [4]

*4) * CALIC – Context Based Adaptive Lossless Image Coder*

Neste tipo de algoritmo, numa primeira fase usa um gradiente baseado numa previsão não linear para obter uma

"imagem perda" e a "imagem residual". Na segunda etapa, o algoritmo usa codificação aritmética.

*5) * LOCO-I – Low Complexity Lossless Compression for Images*

Baseia-se em um modelo de contexto fixo simples, que aborda a capacidade das técnicas universais mais complexas para capturar dependências de alta ordem. O modelo é ajustado para o desempenho eficiente conjuntamente com uma família prolongada de códigos do tipo Golomb, que são escolhidos adequadamente, e uma extensão do alfabeto embebido.

*6) * FELICS - Fast Efficient & Lossless Image Compression System*

Usa os dois vizinhos mais próximos de um pixel para obter diretamente uma distribuição de probabilidade aproximada para a sua intensidade, em combinação com a modelagem de previsão e erro. Usa uma técnica para selecionar o mais próximo de um conjunto de modelos de erro, cada um correspondente a um código prefixo simples. Finalmente codifica a intensidade usando o código prefixo selecionado.

7) LZ-77 Encoding

É um algoritmo de compressão baseado em dicionários, LZ77 funciona explorando a repetição de palavras e frases num arquivo. Um conceito da janela deslizante é usado para fazer referência a dados lidos anteriormente. Se uma palavra ou frase é repetida, um ponteiro de referência é definido para a ocorrência anterior a essa palavra/frase. [5]

8) Deflate

Usa tanto o LZ77 como a codificação de Huffman para a compressão.

9) LZMA - Lempel Ziv Markov Chain Algorithm

LZMA fornece alta taxa de compressão ao comprimir bytes desconhecidos. O algoritmo LZMA usa uma janela deslizante (técnica de compressão do algoritmo LZ77) juntamente com um filtro Delta e Range Encoder.

10) Codificação Aritmética

A partir de um modelo estatístico, constrói-se uma tabela onde são listadas as probabilidades de o próximo símbolo lido ser cada um dos possíveis símbolos.

*11) * Huffman Modificado*

Utilizado para codificar imagens a preto e branco, combina os códigos de comprimento variável da codificação de Huffman com a codificação de dados repetitivos utilizando RLE.

12) DPCM - Differential Pulse Code Modulation

Utiliza uma transformação para aumentar a compressibilidade de uma imagem. Envolve a digitalização da imagem e a previsão do próximo valor de pixel. Existem vários modos para prever o próximo valor de pixels. A média dos pixels à esquerda e o pixel acima é usado como o valor previsto. O conjunto de diferenças entre cada pixel e seu valor previsto é a imagem residual. A distribuição residual é mais compacta que a imagem original. Isso resulta em uma entropia mais baixa, que determina o comprimento mínimo da palavra de código.

13) PPM – Prediction by Partial Matching

Conta o número de vezes que cada símbolo ocorreu anteriormente, e em que contexto, atribuindo-lhe uma probabilidade de acordo, sendo essa usada, depois, num codificador aritmético adaptativo. Desta forma é possível atribuir uma probabilidade ao símbolo que depende não só da frequência absoluta, mas também do contexto em que ocorre.

14) DMC – Dynamic Markov compression

Usa *arithmetic coding* preditivo semelhante a *PPM*, com a diferença que o input é preditado 1 bit de cada vez (em vez de 1 byte).

D. Aplicações (formatos / codecs)

1) PNG - Portable Network Graphics

Apresenta dois estádios – prediction e o deflate. Antes da aplicacação do deflate, os dados são transformados através de um método de previsão (prediction): um método de filtro é usado para toda a imagem, enquanto para cada linha de imagem, um tipo de filtro é escolhido para transformar os dados para torná-lo mais eficientemente compressível.

2) TIFF/TIF - Tagged Image File Format

TIF lida com vários esquemas de compressão, e por isso recorre a tags para armazenar a informação acerca do método de compressão utilizado. Essas opções são, principalmente, além de nenhum método, LZW, PackBits (RLE) e CCITT Group 3 1-Dimensional Modified Huffman (Huffman Modificado).

3) GIF - Graphics Interchange Format

Usa apenas o LZW.

4) Lossless JPEG - Joint Photographic Experts Group

Utiliza um algoritmo de previsão (prediction: DPCM) baseado nos três pixéis mais próximos (em cima, esquerdo e o superior esquerdo), e a codificação da entropia (a codificação Huffman ou a codificação aritmética). [3] [4]

a) JPEG-LS

O JPEG-LS é sustentado pelo algoritmo LOCO-I, que se baseia em previsão, modelagem de resíduos e codificação baseada em contexto dos resíduos. Além da compressão sem perdas, o JPEG-LS também oferece uma opção com perdas, mas "quase sem perdas", onde o erro absoluto máximo pode ser controlado pelo codificador.

b) JPEG 2000

Inclui um filtro de wavelet inteiro especial.

c) JPEG XT

O JPEG XT inclui um modo de transformação DCT (sem perdas) com base na compactação wavelet do JPEG 2000.

5) HEIF - High Efficiency Image File Format

Possibilidade de usar compressão sem perdas e com perdas – baseado no HEVC (ou seja, LZMA, Huffman, DPCM, entre outros).

6) Lossless BPG - Better Portable Graphics

Baseado no HEVC (ou seja, LZMA, Huffman, DPCM, entre outros). [6]

7) JBIG – Joint Bi-level Image Experts Group

Imagens apresentam apenas duas cores. É baseado numa forma de codificação aritmética desenvolvida pela IBM (conhecida como codificador Q) que também usa um refinamento menor desenvolvido pela Mitsubishi, resultando no que ficou conhecido como codificador QM. Baseia as estimativas de probabilidade para cada bit codificado nos valores dos bits anteriores e nos valores nas linhas anteriores da imagem. (ou seja, prediction). [7]

8) Lossless WebP

O formato usa imagens de sub-resolução, recursivamente incorporadas ao próprio formato, para armazenar dados estatísticos sobre as imagens, como códigos de entropia usados, preditores espaciais, conversão de espaço de cores e tabela de cores. LZ77, codificação de Huffman e um cache de cores são usados para a compactação. [8]

9) FLIF - Free Lossless Image Format

Transformações de cores, passagem pela imagem e previsão de pixéis (prediction) com Adam Interlacing, e codificação da entropia: MANIAC (variante do CABAC). [9]

10) 7-Zip

Recove a diversos algoritmos (sendo PPM um deles) para compactar arquivos para o formato 7z, atingindo bons rácios de compactação.

11) Bzip2

Usa diferentes tipos de técnicas "umas em cima das outras", nomeadamente RLE, Burrows-Wheeler transform, Move-to-Front transform e Huffman, resultando em arquivos do tipo .bz2.

III. Aplicação: Compressão para Imagem

Iniciou-se por aplicar os diferentes codecs apresentados anteriormente na imagem fornecida e obtivemos os seguintes resultados:

Tamanho Original (bytes)	33 987 318	
Formato/Codec	**Tamanho Final (bytes)**	**Taxa de Compressão**
PNG	7 559 822	4,496
TIFF	11 193 430	3,036
JPG	4 335 689	7,839
WebP	2 232 062	15,227
HEIC	2 119 374	16,036

Fig. 2. Comparação entre codecs face a taxa de compressão.

Verificou-se, face aos valores na tabela, que codecs que utilizavam a seguinte configuração: Prediction + LZMA/LZMA2/XZ (algum dicionário) + Huffman, tinham taxas de compressão melhores.

Face ao código disponível, utilizável e redirecionado para este contexto, não foi utilizado Prediciton.

Considerou-se, também, utilizar o RLE, mas sem criar redundância primeiro, não fazia sentido, pois é difícil numa imagem com um espectro de cores (cinza) encontrar valores iguais e seguidos numa linha de pixéis.

Optou-se por usar apenas XZ + Huffman. XZ, porque utiliza o LZMA e o LZMA2, sendo dicionários melhorados de outras técnicas. O Huffman para a codificação da Entropia.

Depois de testes e considerando os valores da taxa de compressão, o código de Huffman não melhorava o resultado final pelo que apenas se considerou o XZ (*).

Código Fonte retirado da Internet ([11]*).

Resultado Final:

Tamanho Final (bytes)	**Taxa de Compressão**
8 624 440	3,941

Fig. 3. Resultado na utilização do código para XZ (LZMA/LZMA2).

IV. Aplicação: Compressão para Texto

Para compressão do ficheiro 'war_and_peace.txt" foi usado o pograma 7-zip, usando os métodos LZMA, LZMA2, PPMd e BZip2, obtendo os melhores resultados de compressão com PPMd.

Tamanho Original (bytes)	3 359 549	
Formato/Codec	**Tamanho Final (bytes)**	**Taxa de Compressão**
PPMd	764 213	4,396
LZMA	935 610	3.591
LZMA2	935 761	3.590
bZip2	888 372	3.782

Fig. 4. Utilização de vários técnicas disponíveis no 7-zip.

V. Notas e Trabalho Futuro

O trabalho foi realizado na perspetiva de maximizar a compressão (sem perdas), ignorando as velocidades de compressão para a escolha da técnica.

No futuro, devia-se utilizar mais do que um ficheiro em cada contexto (imagens em escala de cinza num e no outro texto em ASCII) para depois calcular a média dos resultados em cada contexto, e assim obter uma conclusão mais real e fiável. Além disso, considerar outras técnicas/codecs que não foram referidas/utilizadas.

Referências

[1] Visvesvaraya National Institute of Technology, "Image compression", https://www.slideshare.net/pareshkamble/image-compression-12093925 , visitado em 19/11/2019;

[2] "Lossless Definition", https://techterms.com/definition/lossless , visitado em 18/11/2019;

[3] Shivangi Saxena, "Image compression", https://www.slideshare.net/shivangisaxena566/image-compress , acedido 20/11/2019;

[4] Shilpa Ijmulwar, Deepak Kapgate, "A Review on - Lossless Image Compression Techniques and Algorithms", International Journal of Computing and Technology, Vol.1, Issue 9, Outubro 2014;

[5] Apoorv Gupta, Aman Bansal, Vidhi Khanduja, "Modern Lossless Compression Techniques: Review, Comparison and Analysis", IEEE, 2017

[6] S. Sahni, B. Vemuri, F. Chen, C. Kapoor, C. Leonard, J. Fitzsimmons, "State of The Art Lossless Image Compression Algorithms", https://www.researchgate.net/publication/2337322_State_of_The_Art_Lossless_Image_Compression_Algorithms , acedido a 20/11/2019

[7] Ville Kyrki , "JBIG Image Compression Standard", https://www.researchgate.net/publication/2352806_JBIG_image_compression_standard , acedido a 20/11/2019

[8] Google, Developers – "Webp Lossless Bitstream Specification" https://developers.google.com/speed/webp/docs/webp_lossless_bitstream_specification , acedido a 20/11/2019

[9] Jon Sneyers, Pieter Wuille, “FLIF: Free Lossless Image Format Based on Maniac Compression”, https://flif.info/papers/FLIF_ICIP16.pdf , acedido a 21/11/2019

[10] http://multimedia.ufp.pt/codecs/compressao-sem-perdas/codificacao-estatistica/codificacao-aritmetica/

[11] Codigo-Fonte (*) para uma parte do Projeto – Imagens, https://tukaani.org/xz/java.html , acedido a 21/12/2019