UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Faculdade de Ciências - Campus Bauru

DEPARTAMENTO DE COMPUTAÇÃO

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MÉTODOS DE COMPRESSÃO DE IMAGEM

BAURU
2018

HUGO CICARELLI

MÉTODOS DE COMPRESSÃO DE IMAGEM

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Orientador: Prof. Dra. Simone das Graças Domingues Prado

Universidade Estadual Paulista "Júlio de Mesquita Filho"

Faculdade de Ciências

Ciência da Computação

BAURU

2018

Hugo Cicarelli
Métodos de Compressão de Imagem/ Hugo Cicarelli. – Bauru, 2018-
23 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dra. Simone das Graças Domingues Prado

Trabalho de Conclusão de Curso – Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Ciência da Computação, 2018.

1. Image Compression 2. Lossless 3. Lossy 4. Open Source I. Prof. Dra. Simone das Graças Domingues Prado. II. Universidade Estadual Paulista "Júlio de Mesquita Filho". III. Faculdade de Ciências. IV. Métodos Compressão de Imagem

Hugo Cicarelli

Métodos de Compressão de Imagem

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Prof. Dra. Simone das Graças Domingues Prado
Orientador

Prof. Dra. Simone das Graças Domingues Prado

Prof. Dra. Andrea Carla

Bauru 2018

Espaço destinado à dedicátoria do texto.

Agradecimentos

Espaço destinado aos agradecimentos.

Espaço destinado à epígrafe.

Resumo

Espaço destinado à escrita do resumo.

Palavras-chave: Palavras-chave de seu resumo.

Abstract

Abstract area.

Keywords: Abstract keywords.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Compressão de imagem permite reduzir seu tamanho em disco, buscando a menor perda de qualidade possível. A busca por isso se dá, nos dias de hoje, não somente pelo fator de utilização de espaço em disco, como também para facilitar acesso a navegação de sites e também sua rapidez. Quanto menor for o tamanho de um arquivo de imagem, menos dados serão requisitados, consumindo menores dados de tráfego. Esse último caso é um ponto importante, pois o crescimento de acesso a websites e aplicativos em dispositivos móveis tem crescido bastante nos útilmos anos. De acordo com o site Statista.com (https://www.statista.com/topics/779/mobile-internet/ - acessado em 27/05/2018), em 2018, o tráfico de internet acessado por celulares foi de 51.2% em todo o globo. Tendo isso em mente, ao acessar um site contendo uma imagem em seu tamanho original, pode-se levar um tempo 5 vezes maior do que ao comprimi-la.

O tópico sobre compressão de imagens tem ganhado grande destaque por conta de perfomance, como também no quesito de armazenamento em disco. Para sanar a necessidade de falta de espaço em disco surgiram novas tecnologias, que é o caso dos Serviços em Nuvem. Esses Serviços permitem que sejam alocados recursos computacionais, cobrando um valor em cima do que se é realmente utilizado. Embora isso sane em parte o problema de armazenamento, ainda se é investido tecnologias e tempo para otimizar arquivos de imagens, pois reflete em um maior número de imagens em um servidor na nuvem pelo mesmo preço.

1.1 Problema

Como comentado, a busca por esse tópico se dá por basicamente dois motivos: acessos mais rápidos à sites na Internet, como também menor utilização de espaço em disco.

Embora que nos dias atuais, HD's apresentam uma capacidade de armazenamento acima dos Terabytes, isso não é suficiente ainda assim. Embora que, de acordo com **??** (**??**), 1 TB seja equivalente a 782.177 disquetes ou, 1.498 CD's, nos dias atuais, com a grande quantidade de consumo e armazenamento de dados, seria questão de dias até que o espaço fosse totalmente preenchido. A existência de Ambientes na Nuvem, que são sistemas remotos os quais se é possível alocar recursos computacionais a medida que se é necessário, ainda assim é desvantajoso não aproveitar totalmente o potencial de arquivos otimizados, pois se é cobrado pelo valor dos recursos utilizados.

Outro problema abordado é o fato de que arquivos de imagens consomem uma grande parte da banda, ao se carregar um site na Internet. Acessos por celulares nem sempre são feitos através de redes de internet WiFi, sendo muitas das vezes sendo acessados pela rua

através de redes 3G/4G. Isso se torna um problema no consumo de dados, ao fazer requisição para arquivos muito grandes, além da demora para ser mostrada uma imagem, muitas vezes renderizadas em dimensões muito menor que a original.

Existem métodos, os quais iremos abortar, que tratam a otimização buscando sempre a menor perda de qualidade, ou dimensões da imagem. Seu uso irá variar com o que o usuário final busca.

2 Objetivos

2.1 Objetivos Gerais

Têm-se como objetivo final a criação de um aplicativo de Compressão de Imagens, o qual irá oferecer para o usuário o melhor cenário para o que se deseja, seja qualidade da imagem ou redimensionando-a.

2.2 Objetivos Específicos

a) Aprender metodologias de compressão de imagem;

b) Aprender algoritmos que permitem implementar as metodologias apresentadas;

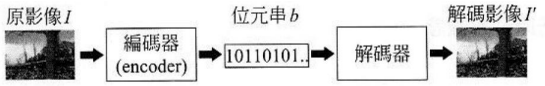
c) Melhorar habilidades com a linguagem PHP;

3 Fundamentação Teórica

Em Ciência da Computação, compressão de dados é o processo de codificar as mesmas informações usando um número menor de bits, sem que haja distorção dos dados originais. Em se tratando de compressão de imagens, pode-se alcançar Esse processo é útil, pois reduz o consumo de recursos computacionais, como espaço em disco, ou utilização de banda de internet.

Considere as imagens abaixo.

Figura 1 – Codificador e Decodificado de Imagens



Fonte: Elaborada pelo autor.

Quando o sistema recebe a imagem original, ele manda para um codificador que converte a imagem original para um fluxo de bits. Um decodificador então recebe esse fluxo de bits e o transforma novamente na imagem. Caso o fluxo de bits final seja menor que o original, chamamos esse processo de Compressão de Imagem.

RAO diz que durante a compressão de uma imagem, pode-se ter perda de dados durante o processo. Por essa razão, o decodificador não consegue reconstruir a imagem perfeitamente para seu estado inicial. Esse tipo de compressão é chamado de Lossy, e esse processo é irreversível. O processo no qual a imagem pode retornar ao seu estado inicial é chamado de *Lossless*, o qual é possível reduzir o tamanho em disco, sem ter perda de dados durante o processo, sendo esse um processo reversível.

3.1 Métodos de Compressão de Imagem Sem Perda

São os chamados tipos de compressão *Lossless*. Neles, como dito anteriormente, é possível restaurar todos os dados originais da Imagem ao descompacta-la. Esse tipo de compressão é geralmente usado para arquivos de imagem de extensão GIF, formato usado na internet o qual são permitidas animação de imagens e utilização de transparência.

De acordo com VEMURI et al. define que compressão de imagens do tipo *Lossless*, também conhecidas como Compressão de Imagem Sem Perda, podem ser alcançadas através de Métodos de Codificação, Domínio Espacial, Domínio de Frequência, os quais são baseados em Transformadas, são uma combinação desses métodos.

Métodos de codificação são diretamente aplicados a imagens de formato RAW, o qual é o formato mais puro de uma imagem, normalmente sendo gerado a partir de câmeras profissionais. Esse formato acumula todos os dados necessários da imagem, como abertura da câmera, tipo da câmera, ISO utilizada, entre outras informações. O método de codificação trata a imagem como uma sequência de números discretos. Métodos comuns dessa incluem aritmética, Huffman, Lempel-Ziv Welch e Run-Length.

Métodos de domínio espacial são uma combinação de algoritmos de domínio espacial e métodos de codificação. Eles não somente operam diretamente nos tons de cinza, que são tons atribuidos a um pixel sendo 0 equivalente a branco e 100 equivalente a preto, como também tentam eliminar a redundância espacial. Essa redundância consiste na semelhança de pixels adjacentes de uma imagem. **??** dá o seguinte exemplo: uma imagem de um avião passando no céu sem nuvens, na qual a informação relevante a ser transmitida é o avião, e o fundo é a parte da imagem azul cujo conteúdo da imagem é praticamente uniforme.

Na compressão de domínos de frequência, a imagem é representada usando uma base apropriada, com o objetivo de se obter um coeficiente de matriz pequeno. Transformada do Cosseno Discreto (DCT) e Transformada Wavelet são exemplos de compressão de domínio de frequência.

3.1.1 Performance

Perfomance em algoritmos do tipo *Lossless* pode ser especificados em termos de complexidade e eficiência.

A complexidade de um algoritmo de compressão de imagem é medida pelo número de operações aritméticas necessárias para realizar ambos processos de codificação e decodificação. Esse é um fator importante para aplicações que envolvem compressão online, onde a velocidade é crucial.

Eficiência de compressão é medida pela proporção de compressão ou pela taxa de bits. Proporção de compressão é o número de bits por pixel de uma imagem comprimida. Por exemplo, se uma imagem de dimensões 256x256 de 8 bits por pixel, é necessários 256 * 256 * 8 bits = 65.536 bytes quando armazenada em sua forma original. Se a imagem otimizada possuir 32.768 bytes, então a proporção de compressão será 65536 / 32768 = 2. Como a imagem possui dimensão de 256x256 = 65.536 pixels, o arquivo comprimido precisa de 32768 * 8 / 65536 = 4 bits, o qual será a taxa de bits necessária. A proporção de compressão portanto está associada a taxa de bits necessária. Sendo *CR* a proporção de compressão, *BR* a taxa de bits e *v* o número de bits por pixel de uma imagem não otimizada, temos a seguinte fórmula:

$$CR = b/BR$$

3.1.2 Métodos de Codificação

Nessa seção, serão abordados alguns algoritmos de codifição, bem como a explicação do método.

Para casos de codificação na qual uma imagem em duas dimensões será comprimida, existe a necessidade de convertê-la para uma sequência de uma dimensão. Para esse processo, chamamos de Linearização.

3.1.2.1 Linearização

A linearização não afeta a frequência da codificação, sendo esse método aplicado para alguns casos. Como Huffman depende somente da frequência dos diferentes tons de cinza, o método não é afetado pela linearização. Agora métodos como Liv-Zempel dependem da ordem dos tons de cinza, sendo então afetados por métodos de linearização.

Imagens possuem o que é chamado de redundância local, o que causa uma certa região da imagem a exibir uma coerência ou correlação, resultando em uma suavidade entre os pixels. Alguns métodos de linearização são mais efetivos que outros em se tratando de preservar essa região e, por isso, são esperados terem melhor desempenho quando combinadas com métodos de codificação que se utilizam dessas regiões de redundância.

Abaixo está uma lista dos métodos de linearização mais utilizados, segundo VEMURI et al.:

a) Verificação orientada por linha (*Row-Major Scan*): a imagem é verificada linha por linha, sentido cima-esquerda para baixo-direita;

b) Verificação orientada por coluna (*Column-Major Scan*): a imagem é verificada coluna por coluna, com sentido cima-esquerda para baixo-direita;

c) Verificação orientada por diagonal (*Diagonal Scan*): a imagem é verificada em diagonais, começando do canto inferior esquerdo para o canto superior direito;

d) Verificação em formato de cobra (*Snake-like row-major Scan*): é uma variação da verificação orientada por linhas e orientada por colunas. Nela, ao chegar no final de uma linha, segue para a linha de baixo continuando na mesma coluna;

e) Verificação em espiral (*Spiral Scan*): nesse método,;

f) Verificação de Peano-Hilbert (*Peano-Hilbert Scan*): essa verificação requer que a imagem seja

$$2^k * 2^k$$

. Quando *k* é ímpar, o caminho percorrido começa no pixel mais a esquerda da primeira linha e termina no pixel mais a esquerda da última linha. Quando *k* é par, o caminho começa no pixel mais a esquerda da primeira linha e termina no pixel mais a direita da última linha.;

Segundo o autor, embora que nenhum método de linearização forneça a melhor compressão, a verificação de Peano-Hilbert geralmente traz o melhor resultado.

3.1.2.2 Codificação de Huffman

Nesse método, a redundância de codificação é eliminada com base numa codificação que produz um código de tamanho variável, atribuindo os códigos de tamanhos menores aos níveis de cinza mais prováveis de ocorrer.

Esse método possui duas etapas:

a) Cria-se uma série de reduções dos símbolos através da junção dos dois de menores probabilidades a cada iteração.

b) Codificam-se todos os símbolos que foram reduzidos, começando com o de maior probabilidade que será associado ao menor código e voltando para os originais.

COMPUTAÇÃO... dá o seguinte exemplo: imagem de tamanho 10x10 e 6 tons de cinza (a1, a2, a3, a4, a5, a6), tendo as seguintes probabilidades de ocorrência: 5/8 de a1, 3/32 de a2 e a3, 1/32 de a6 e a4, e 1/8 de a5.

Figura 2 – Primeira etapa de codificação de Huggman.

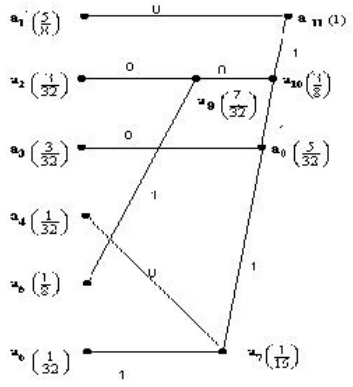


Tabela referente a segunda etapa da codificação de Huffman para as probabilidades das palavras mostradas na figura anterior.

Para transmitir essa informação, obtem-se uma taxa média de bits/informação seguindo a seguinte fórmula:

$$(5/8) * 1 + (3/32) * 3 + (3/32) * 3 + (4/32) * 3 + (1/32) * 4 + (1/32) * 4 = 1,813$$

bits/informação.

Tabela 1 – Segunda etapa da codifiçãode Huggman

Informação	Probabilidade	Código
a1	$5/8 = 20/32$	0
a10	$3/8 = 12/32$	1
a9	$7/32$	10
a8	$5/32$	11
a5	$1/8 = 4/32$	101
a2	$3/32$	100
a3	$3/32$	110
a7	$2/32$	111
a4	$1/32$	1110
a6	$1/32$	1111

3.1.2.3 Codificação por LZW

Lempel, Ziv e Welch propuseram um método de codificação adaptável que não requer que todos que irão ser compressados, estejam disponíveis desde o começo. Essa técnica gera o código conforme o vai examinando, do começo ao fim. Na codificação por Huffman, um código de tamanho variável é construído para cada símbolo no código fonte. Na codificação Lampel-zev, códigos de tamanho fixo são construídos a medida que o processo vai rodando, para cada sequência de símbolos de tamanho variável.

Suponhamos que os símbolos que ocorrem em um código seja *a*, *b*, e *c*, e que a cadeia de caracteres *ababcabc* está para ser comprimida. Primeiro, iniciamos o dicionário para essa tradução no qual apenas um símbolo seja possível. Como o código do dicionário é dado pela posição, *a* seria 0, *b* seria 1 e *c* seria 2. Ao ler a cadeia de caracteres ababcabc, temos o primeiro sendo *a*. Seu código 0 seria parte do arquivo comprimido em conjunto com o prefixo e a próxima entrada. Nesse caso, seria *ab* e o código seria 3. A parte que restou para compressão agora seria *babcabc*. Novamente pegamos a cadeia de maior prefix que sobrou, sendo *b* nesse caso. Seu código é 1, somando com o próximo prefixo que é *a*, temos o código 4. Como agora temos no dicionário os caracteres *a* e *b*, a próxima sequência que pegaremos da cadeia restante, *abcabc* será *ab*. O código dela é 3, somando com o código do próximo caracter *c*, temos o valor 5. Como próxima sequência encontrada é *c*, com código 2 e temos *ca* entrando no dicionário. A cadeia restante então é *abc*, a qual está no dicionário, com valor 5. Com isso, foi possível chegar na codificação da cadeia *ababcabc*, a qual é 01325.

O método de compressão gzip, utilizado por sistemas Unix, utiliza uma combinação dos métodos de Huffman e LZW.

3.1.2.4 Codificação por Código de Tons Corridos (RLE)

Nesse método, o código fonte é dividido em segmentos de símbolos idênticos. Cada segmento é separado por um símbolo e o número de ocorrências.

Para elucidar, suponha a cadeia *aaaabaaabb*. Ela é codificada como (a, 4), (b, 1), (a, 3), (b, 3). Esse método de codificação funcional bem para cadeias que possuem segmentos grandes, como imagens com fundos uniformes, porém esse método não é tão eficaz quando as cadeias possuem muitos segmentos curtos

3.1.3 Métodos de Domínio Espacial

HANSAN et al. que os algoritmos de domínio espacial envolvem métodos para reduzir o número de bits que representam a informação contida na imagem operando diretamente em seu formato mais cru, o qual contém maior número de informações. Segundo

3.1.4 Métodos de Domínio de Frequência

4 Metodologia

4.1 Métodos e Etapas

Para o desenvolvimento do projeto foi realizado o levantamento de tecnologias atualmente existentes que podessem proporcionar o desenvolvimento da ferramenta. Por ter se planejado em ser uma aplicação Web, foi levado em consideração tecnologias para tal.

Com as tecnologias definidas, reuniu-se uma base teórica referentes à compressão de imagens, buscando ver novos métodos e quais seriam aplicados no desenvolvimento, bem como diferenciando diversos métodos já existentes com seus pontos positivos e negativos.

A última etapa do processo foi a de desenvolvimento da aplicação.

4.2 Materiais Utilizados

4.2.1 Ambiente de desenvolvimento

Para desenvolvimento do projeto foi utilizado o Sistema Operacional Windows 10, tendo em vista que o Servidor o qual a aplicação seria hospedada seria um ambiente Linux, distribuição Ubuntu.

4.2.2 Github

O Github é ao mesmo tempo um servidor de armazenamento de código e uma rede social onde pode-se submeter modificações, fazer cópias e acompanhar modificações de códigos de outras pessoas. A rede foi essencial para o desenvolvimento deste projeto por armazenar vários sub-módulos e disponibilizar código-fonte para consulta.

4.2.3 TinyPNG

TinyPNG é um aplicativo online semelhante à proposta desse projeto. Nele, é possível otimizar imagens de graça, com limite de 500 imagens/mês em uma conta gratuita. Esse aplicativo online foi utilizado para comparar o resultado da aplicação final.

4.2.4 Codeigniter

Codeigniter é um framework para a linguagem PHP. Frameworks facilitam o desenvolvimento pois garantem uma aplicação estruturada, preparada para fácil manutenção e atualização.

Além disso, são mais rápidos pois permite que códigos e bibliotecas sejam reaproveitadas, de modo que o programador foca apenas no desenvolvimento do módulo pretendido.

Referências

COMPUTAÇÃO Gráfica - Vol. 2 - Cap. 8 - Compressão de Imagem. <http://computacaografica.ic.uff.br/transparenciasvol2cap8.pdf>. [Acessado em 18 de Agosto de 2018]. Citado na página 18.

HANSAN, M. et al. *Spatial Domain Lossless Image Compression Technique by Reducing Overhead Bits and Run Length Coding*. [S.l.]: International Journal of Computer Science and Information Technologies, 2012. ISBN 09759646. Citado na página 20.

Capítulo 3 - processos de compressÃo. In: RAO, V. B. C. Y. (Ed.). *Revista Mackenzie de Engenharia e Computação*. Ano 5. [s.n.]. p. 34 – 43. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9780123736246500059>. Citado na página 15.

VEMURI, B. C. et al. *Lossless Image Compression*. <https://www.researchgate.net/profile/Sartaj_Sahni/publication/2555508_Lossless_Image_Compression/links/54340d4a0cf2bf1f1f27b3d4.pdf>. [Acessado em 18 de Agosto de 2018]. Citado 2 vezes nas páginas 15 e 17.