Universidade Federal de Santa Catarina
Centro Tecnológico
Departamento de Informática e Estatística
Ciência da Computação
INE5431-07208 - Sistemas Multimídia

Relatório - Prática 3

Guilherme Augusto Oliveira Pedrozo (22100621)
Julia Macedo de Castro (23250860)
Victor Henrique Labes de Figueiredo (22200378)

Florianópolis
2024

Questão 1. Abra o arquivo peixe.bmp no editor hexadecimal em https://hexed.it/ e, analisando o formato do cabeçalho BMP apresentado na Seção 2, indique no relatório: qual é o valor dos campos offset e tamanho do arquivo? Quais são os valores dos componentes de cor R, G e B do primeiro pixel armazenado no arquivo?

Questão 2. Qual é o tamanho do cabeçalho do arquivo peixe1.cuif para seu grupo?

Questão 3. No arquivo praticaIII.py tem um função PSNR incompleta. Implemente esta função de maneira a calcular o PSRN passando como parâmetro a imagem original e uma decodificada. Implemente o cálculo do MSE e PSNR com base nas fórmulas da seção 5 .

Questão 4. Indique o PSNR comparando a imagem original peixe.bmp (original) com a imagem obtida a partir do arquivo CUIF.1 (peixe1.bmp). Justifique a resposta do PSNR.

Questão 5. Compacte as imagens peixe.bmp e peixe1.cuif com zip. Qual a taxa de compressão obtida para os dois arquivos? Qual arquivo compactou mais? Explique porque deste resultado, ou seja, indique a vantagem de organizar os pixels nesta sequência definida pelo CUIF.1 (primeiro os valores de R, depois de G e finalmente de B) para a compressão baseada em RLE ou DPCM? Dica: relembre os princípios da compressão RLE e DPCM e compare a parte de dados de imagem do arquivo peixe.bmp e peixe1.cuif no editor hexadecimal.

Questão 6. Agora altere o código em PraticaIII.py para que seja gerado o arquivo peixe2.cuif, que utiliza a versão CUIF.2 (usar 2 em vez de 1 para indicação da versão) e peixe2.bmp. Visualiza as imagens peixe.bmp e peixe2.bmp para ver se existem diferenças visíveis. Analise o código que gera o arquivo CUIF.2 (em Cuif.py) e explique o princípio da compressão adotada no CUIF.2

Questão 7. Indique as taxas de compressão obtidas pelos CUIF.1 e CUIF.2 para a imagem peixe.bmp? Para este cálculo determine a razão entre um arquivo cuif e a imagem peixe.bmp. Qual versão do CUIF compactou mais?

Questão 8. Indique o PSNR comparando a imagem original peixe.bmp (original) com a imagem obtida a partir do arquivo CUIF.2 (peixe1.bmp). Justifique a resposta do PSNR.

Respostas:

QUESTÃO 1:

peixe.bmp:

- Offset: 36 00 00 00 (hex.) = 54 (dec.);
- Tamanho do arquivo: F6 72 09 08 (Little Endian) = 15*16^7 + 6*16^6 + 7*16^5 + 2*16^4 + 9*16^3 + 0*16^2 + 0*16^1 + 0*16^0 = 619254 bytes;
- Cores do 1º pixel:
    - Vermelho (R): 91 (hex.) = 145 (dec.);
    - Verde (G): 69 (hex.) = 105 (dec.);
    - Azul (B): 38 (hex.) = 56 (dec.).

OBS.: 'Dec.' = 'decimal', 'hex'. = 'hexadecimal'.

QUESTÃO 2:

- Tamanho do cabeçalho ('peixe1.cuif'): 4*(Assinatura) + 1*(Versão) + 1*(Número de estudantes) + 4*(Largura) + 4*(Altura) + 12*(Matrículas) = 4*1 + 1*1 + 1*1 + 4*1 + 4*1 + 4*3 = 26 bytes

QUESTÃO 3:

```
def PSNR(original,decodificada,b): #### Está correto de acordo com a fórmula, não
                                        # houve alteração.
    try:
        mse = MSE(original,decodificada)
        psnr = 10*math.log10(((2**b-1)**2)/mse)
        return psnr
    except ZeroDivisionError:
        return "Infinito"

def MSE(ori, dec): #Completo - Questão 3 do relatório.
    mse = 0
    nsymbols = ori.width * ori.height * 3
    for i in range(ori.width):
        for j in range(ori.height):
            ori_r, ori_g, ori_b = ori.getpixel((i, j))
            dec_r, dec_g, dec_b = dec.getpixel((i, j))
            mse += (ori_r - dec_r) ** 2
            mse += (ori_g - dec_g) ** 2
            mse += (ori_b - dec_b) ** 2
    mse /= nsymbols
    return mse
                    #Completo - Questão 3 do relatório.
```

QUESTÃO 4:

O PSRN obtido é 'infinito', pois a qualidade de imagem do arquivo original (peixe.bmp) e (peixe1.bmp) são iguais. Logo o MSE entre ambas resulta em 0 e ao calcular o PSRN, resulta em uma divisão por zero.

Portanto, conclui-se que não há erro/ruído na imagem codificada.

QUESTÃO 5:

- Taxa de compressão para ‘peixe1.bmp’:
    - Tamanho original: 619,254 KB;
    - Tamanho compactado: 502,792 KB;
    - Taxa de compressão: (619,254 – 502,792)/ 619,254 = 0,188068224 ~= 0,1881 ~= 18,81%.

- Taxa de compressão para ‘peixe1.cuif’ (usando Windows 11):
    - Tamanho original: 619,226 KB;
    - Tamanho compactado: 486,966 KB;
    - Taxa de compressão: (619,226 – 486,966)/ 619,226 = 0,213589223 ~= 0,2136 ~= 21,36%.

OBS.: Os tamanhos dos arquivos ‘peixe1.bmp’ e ‘peixe1.cuif’ (original e comprimido de ambos) foram obtidos através do comando via terminal Linux: ‘ls -l <nome_do_arquivo>’ .

‘peixe1.cuif’ compactou mais, obtendo uma taxa de compressão maior em relação ao ‘peixe1.bmp’ compactado.

A organização dos pixels em blocos de cores no formato CUIF.1 pode aumentar a chance de valores repetidos aparecerem consecutivamente, facilitando a compressão por meio do RLE, já que há maior probabilidade de sequência de cores semelhantes de dentro de uma mesma componente dentro de um mesmo componente.

Como o DPCM depende da correlação entre pixels vizinhos, a separação das componentes pode criar padrões de diferença previsíveis dentro de cada canal de cor (R, G ou B). Isso permite uma codificação mais eficiente das diferenças entre pixels adjacentes, já que a variação entre as cores dentro um mesmo canal pode ser mais gradual, facilitando a compressão.

QUESTÃO 6:

Comparando as imagens ‘peixe.bmp’ e ‘peixe2.bmp’, é perceptível uma pequena perda de qualidade na imagem ‘peixe2.bmp’.

A compressão adotada no CUIF.2 baseia-se na combinação de componentes de cor verde (G) e azul em único byte, economizando espaço em comparação com o armazenamento não compactado de cores. A combinação consiste na utilização dos 4 bits mais significativos do componente verde (G) e dos 4 bits menos significativos do componente azul (B), ou seja, na função ‘generateCUIF2(self,img)’ cada componente é separado em vetores (‘arrays’) ‘r’ (Vermelho), ‘g’ (Verde) e ‘b’ (azul). E a combinação dos dois componentes é feita pela váriavel ‘gb’, que recebe ‘(b>>4) + (g&F0)’. Logo resultando em um byte comprimido que contém informações de ambos, porém os componentes vermelho (R) são armazenados diretamente, sem compressão.

QUESTÃO 7:

- Taxa de compressão para ‘peixe1.cuif’:
    - Tamanho original: 619,226 KB;
    - Tamanho compactado: 486,966 KB;
    - Taxa de compressão: (619,226 – 486,966)/ 619,226 = 0,213589223 ~= 0,2136 ~= 21,36%.

- Taxa de compressão para ‘peixe2.cuif’:
    - Tamanho original: 412,826 KB;
    - Tamanho compactado: 228,450 KB;
    - Taxa de compressão: (412,826 – 228,450)/ 412,826 = 0,446619157 ~= 0,4466 ~= 44,66%.

A versão CUIF.2 obteve uma compressão significativamente maior, com 44,66% de compressão, em comparação com 21,36% da versão CUIF.1. Logo, CUIF.2 compactou mais.

QUESTÃO 8:

- Cálculo do PSNR (‘peixe1.bmp’ usando CUIF**.**1): Infinito;
- Cálculo do PSNR (‘peixe2.bmp’ usando CUIF.2): 31.008206438788747.

Um PSNR de 31.008 dB é um bom indicativo de qualidade em compressão de imagem. Valores acima de 30 dB geralmente significam que a qualidade da imagem é ainda aceitável para visualizações em tela e algumas aplicações práticas. No entanto, pode haver artefatos perceptíveis se o observador estiver atento. Embora tenha havido alguma perda de qualidade, a imagem ainda preserva uma boa quantidade de detalhes e não apresenta degradações severas. Ao comparar este

PSNR com um PSNR infinito (que indicaria que a imagem original e a decodificada são idênticas), podemos concluir que a técnica de compressão utilizada em CUIF.2 introduziu uma degradação que, embora seja visível, ainda é razoável para muitas aplicações.