Fundamentos de Processamento de Imagem - INF01046 - UFRGS

Txai Durigon Wieser - 00217052
Marcos Vinícius de Oliveira Pinto - 00288560

Algoritmos de Compressão e Análise de Imagens.

Nosso uso de computadores que armazenam quantidades grandes de imagens aumenta a cada dia. Celulares que carregam uma biblioteca fotográfica pessoal, sistemas web que trafegam imagens para mostrar no navegador e muito mais. Compressão de imagens se tornou uma necessidade para que possamos ter eficiência nas transferências de imagens e também em seu armazenamento. Existem hoje muitos métodos de compressão de imagens, que se enquadram em métodos com perda e métodos sem perda.

1) Compressão baseada no modo básico do JPEG:

O processo JPEG é um dos mais usados da categoria com perdas, e funciona utilizando transformadas discretas de cosseno (DCT).

A Transformada discreta de cosseno funciona separando imagens em partes de diferentes frequências. Logo após, aplicamos uma matriz de quantização que realiza a compressão, frequências menos importantes são descartadas. Assim, na hora de descompressão somente as frequências mais importantes serão reconstruídas, e o resultado disso é que teremos menos detalhamento e podem aparecer artefatos de distorção.

De forma alto nível, os passos necessários são:

- Separar a imagem em blocos de 8x8 pixels.
- Aplicar a transformada discreta do cosseno para cada bloco.
- Multiplicar cada bloco pela matriz de quantização relativa a qualidade desejada.
- Essa matriz resultado pode ser armazenada de uma maneira bem mais eficiente.
- Para reconstruir basta realizar o processo da transformada discreta do cosseno inversa (IDCT).

A equação do DCT para calcular entradas i e j de uma matriz é a seguinte:

$$D(i,j) = \frac{1}{\sqrt{2N}} C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} p(x,y) \cos\left[\frac{(2x+1)i\pi}{2N} \right] \cos\left[\frac{(2y+1)j\pi}{2N} \right]$$

$$C(u) = \left\{ \begin{array}{ll} \frac{1}{\sqrt{2}} & \text{if } u = 0 \\ 1 & \text{if } u > 0 \end{array} \right\}$$

Onde p(x, y) é o elemento x-y na matriz p. N é o tamanho do bloco onde vamos aplicar a DCT. Para facilitar podemos gerar uma matriz 8x8 correspondente a essa função, já que operações com matrizes são muito eficientes na computação.

$$T = \begin{bmatrix} .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 \\ .4904 & .4157 & .2778 & .0975 & -.0975 & -.2778 & -.4157 & -.4904 \\ .4619 & .1913 & -.1913 & -.4619 & -.4619 & -.1913 & .1913 & .4619 \\ .4157 & -.0975 & -.4904 & -.2778 & .2778 & .4904 & .0975 & -.4157 \\ .3536 & -.3536 & -.3536 & .3536 & .3536 & -.3536 & -.3536 & .3536 \\ .2778 & -.4904 & .0975 & .4157 & -.4157 & -.0975 & .4904 & -.2778 \\ .1913 & -.4619 & .4619 & -.1913 & -.1913 & .4619 & -.4619 & .1913 \\ .0975 & -.2778 & .4157 & -.4904 & .4904 & -.4157 & .2778 & -.0975 \end{bmatrix}$$

Depois de obtermos essa matriz referente a DCT podemos multiplicar pela matriz de valores da imagem seguindo blocos 8x8. Como a matriz de DCT funciona com valores de pixels de -128 a 127 devemos lembrar de subtrair 128 de cada pixel da imagem antes da multiplicação.

$$D = TMT'$$

Após termos nossa matriz já transformada, vamos realizar a compressão em si usando quantização. Através de uma matriz padrão no JPEG podemos obter outras matrizes para comprimir a imagem usando outras qualidades. A qualidade 50, por experimentos é a que conseguimos uma compressão boa ainda mantendo uma excelente qualidade de imagem depois da descompressão. Portando a Q50 é a nossa matriz de quantização padrão. Se desejarmos uma qualidade maior que 50 devemos multiplicar a matriz de quantização por (100-qualidade)/50. Se quisermos uma compressão maior, ou seja, uma qualidade menor que 50, devemos multiplicar a matriz por (50/qualidade).

$$Q_{50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Para comprimirmos a imagem então realizamos a multiplicação das matriz imagem 8x8 pela matrix de quantização, Arredondando o resultado.

$$C_{i,j} = round\left(\frac{D_{i,j}}{Q_{i,j}}\right)$$

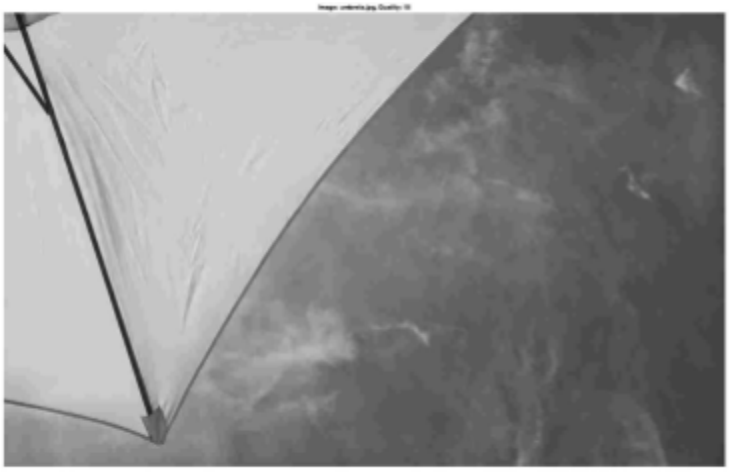
Para resolver essa questão escolhemos duas imagens em tons de cinza e iremos aplicar compressão e descompressão JPEG e analisar os resultados.

[Image: umbrela.jpg, Original]

[Image: umbrela.jpg, Quality: 50]

SNR: 15.423412dB
PSNR = 26.818287dB

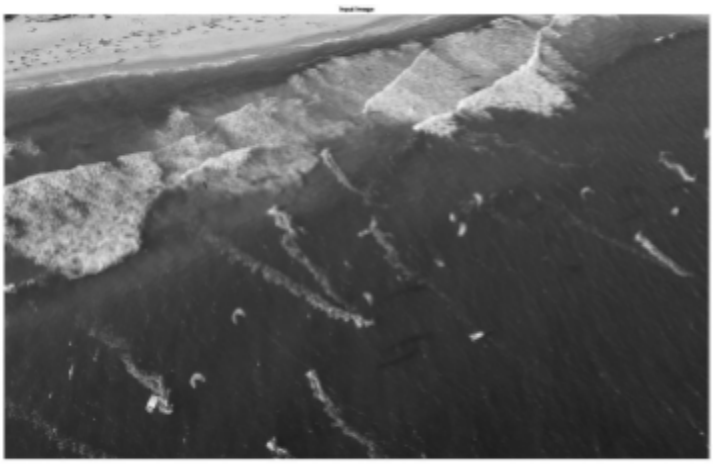
Elapsed time is 3.843708 seconds.



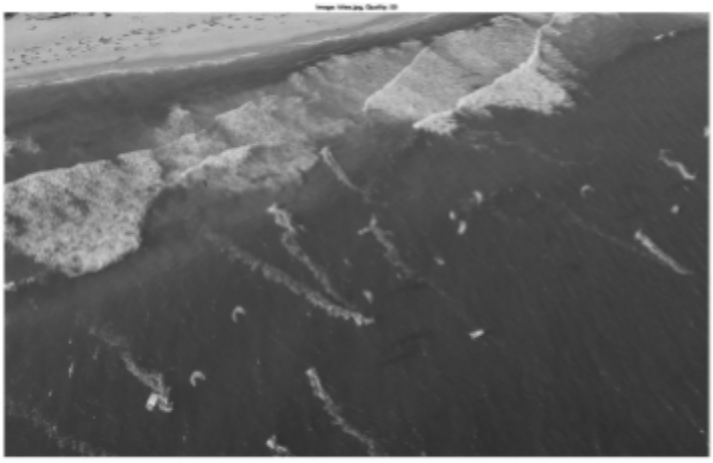
[Image: umbrela.jpg, Quality: 10]

SNR: 12.102152dB
PSNR = 23.638070dB

Elapsed time is 4.075759 seconds.



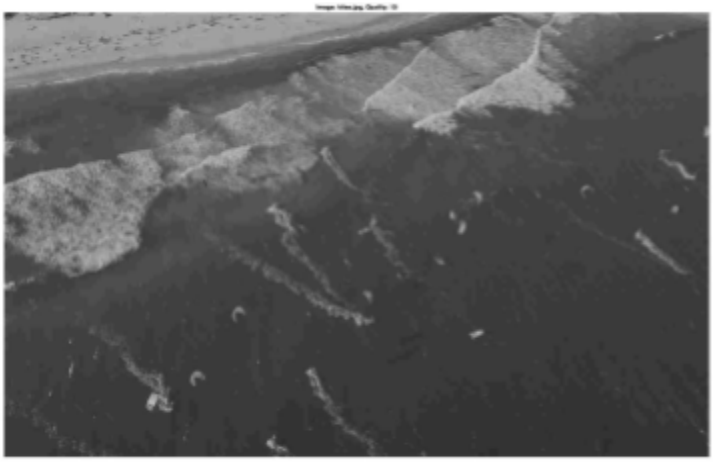
[Image: kites.jpg, Original]



[Image: kites.jpg, Quality: 50]

SNR: 12.369154dB
PSNR = 25.629346dB

Elapsed time is 3.820575 seconds.



[Image: kites.jpg, Quality: 10]

SNR: 8.438101dB
PSNR = 22.711625dB

Elapsed time is 3.823046 seconds.

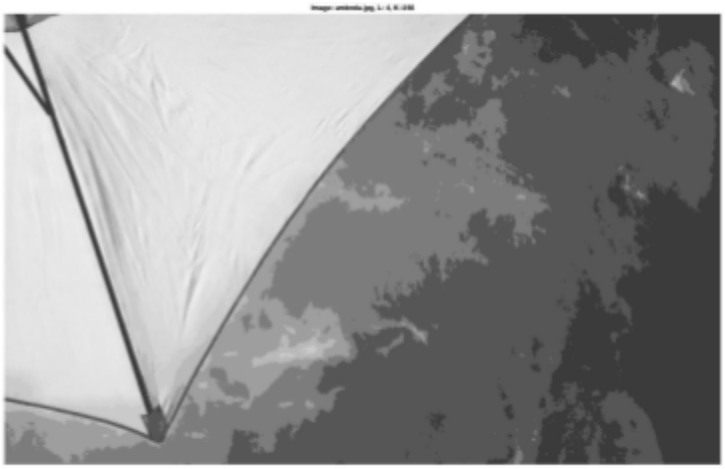
Como podemos ver, com qualidade 50 conseguimos uma ótima compressão da imagem, ainda mantendo quase imperceptíveis perdas de qualidade. Por isso a Q50 virou padrão. Já com qualidade 10 podemos ver que a imagem apresenta perda de definição nas bordas, serrilhamento nas linhas transversais e menos contraste em geral.

2) Compressão baseada na quantização vetorial:

Compressão baseada na quantização vetorial é uma técnica de compressão com perdas. A quantização funciona dividindo um grande conjunto de pixels em grupos com aproximadamente o mesmo número de pontos mais próximos. Cada grupo é representado por um pixel central.

K-means é um algoritmo de clustering que junta pixels baseados no numero de clusters queremos nos nossos dados, no caso K é esse número. O algoritmo seleciona o cluster inicial e baseado na distância de cada pixel faz a atribuição para cada cluster.

[Image: umbrela.jpg, Original]



[Image: umbrela.jpg, L: 4, K: 256]

Input Size = 829440 bytes
Output Size = 208384 bytes

Compression: 3.98 x 1

SNR: 27.815820dB
PSNR = 32.107063dB

Elapsed time is 161.613499 seconds.

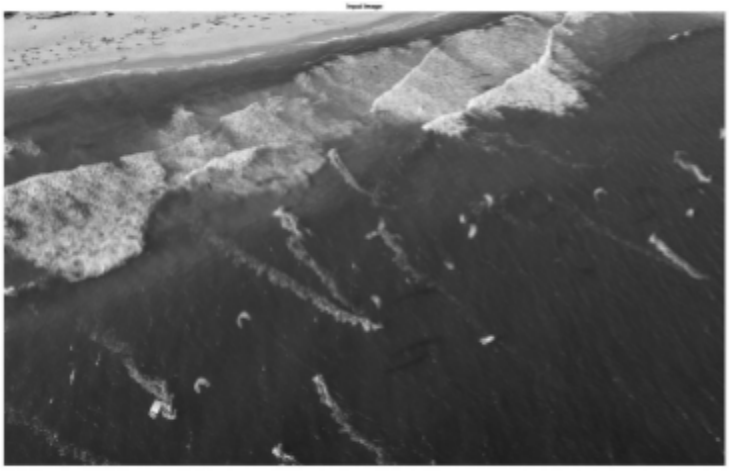
[Image: umbrela.jpg, L: 9, K: 256]

Input Size = 829440 bytes
Output Size = 94464 bytes

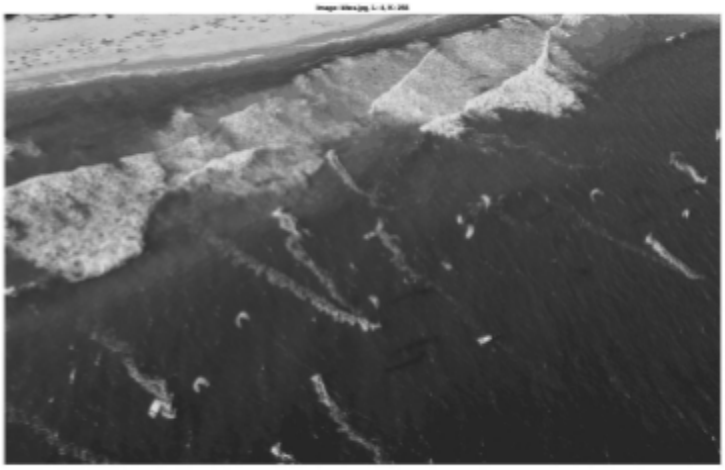
Compression: 8.78 x 1

SNR: 23.795217dB
PSNR = 28.086460dB

Elapsed time is 78.900444 seconds.



[Image: kites.jpg, Original]



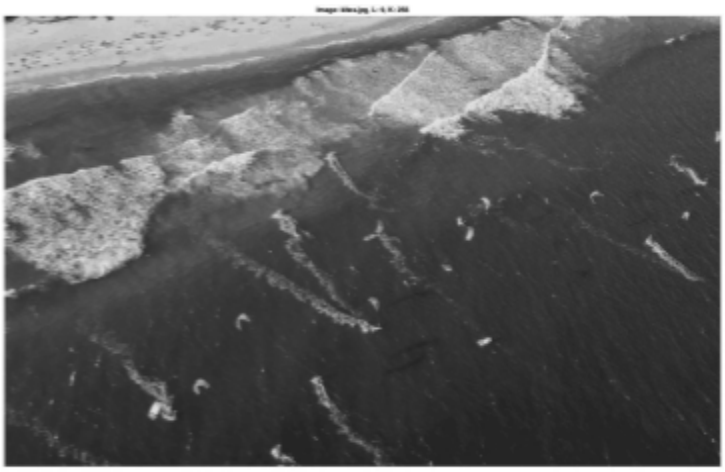
[Image: kites.jpg, L: 4, K: 256]

Input Size = 829440 bytes
Output Size = 208384 bytes

Compression: 3.98 x 1

SNR: 21.216988dB
PSNR = 29.515982dB

Elapsed time is 185.622591 seconds.



[Image: kites.jpg, L: 9, K: 256]

Input Size = 829440 bytes
Output Size = 94464 bytes

Compression: 8.78 x 1

SNR: 20.006881dB
PSNR = 28.305875dB

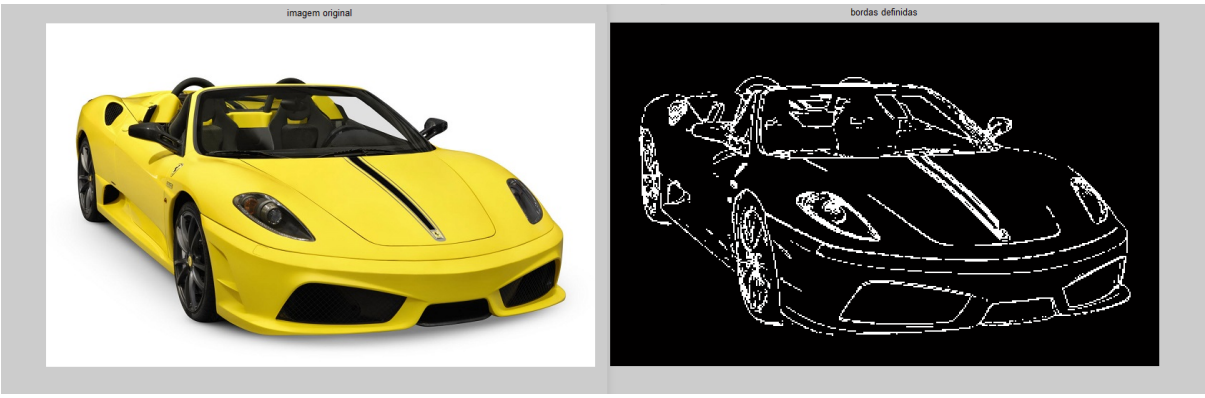
Elapsed time is 78.396019 seconds.

Como podemos ver através dos exemplos acima, quando usamos blocos menores 2x2 temos uma definição maior da imagem, mantendo mais características da imagem original. Já quando vamos aumentando o tamanho dos blocos (3x3) mais detalhes se perdem. Uma característica dessa quantização por clusters é que um grupo de cores originalmente similares apontam para o mesmo cluster assim resultando na mesma cor do pixel. Isso vai formando artefatos onde encontramos gradientes na imagem.

3) Segmentação:

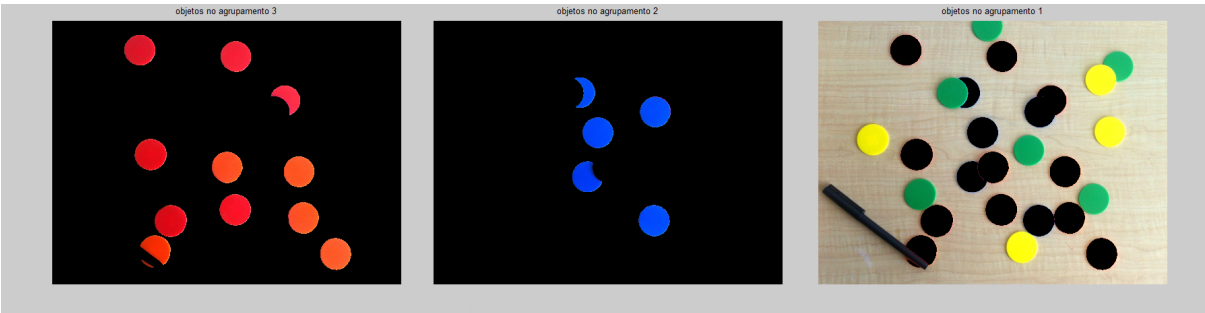
O primeiro método de segmentação escolhido foi o de split-merge, durante o desenvolvimento tudo ocorreu com fluidez e os resultados foram de acordo com o esperado, foi escolhido uma dimensão mínima de tamanho 2 para o encerramento dos splits quadráticos e a função que define se uma região é homogênea utiliza de uma combinação do desvio de valores da região e da média dos valores na mesma região.

Esse método foi escolhido pois nossa imagem à segmentar era de um carro de cor homogênea destacado em um fundo de cor distinta e também homogênea, possibilitando conseguirmos além do contorno externo do carro também seus vários detalhes menores.



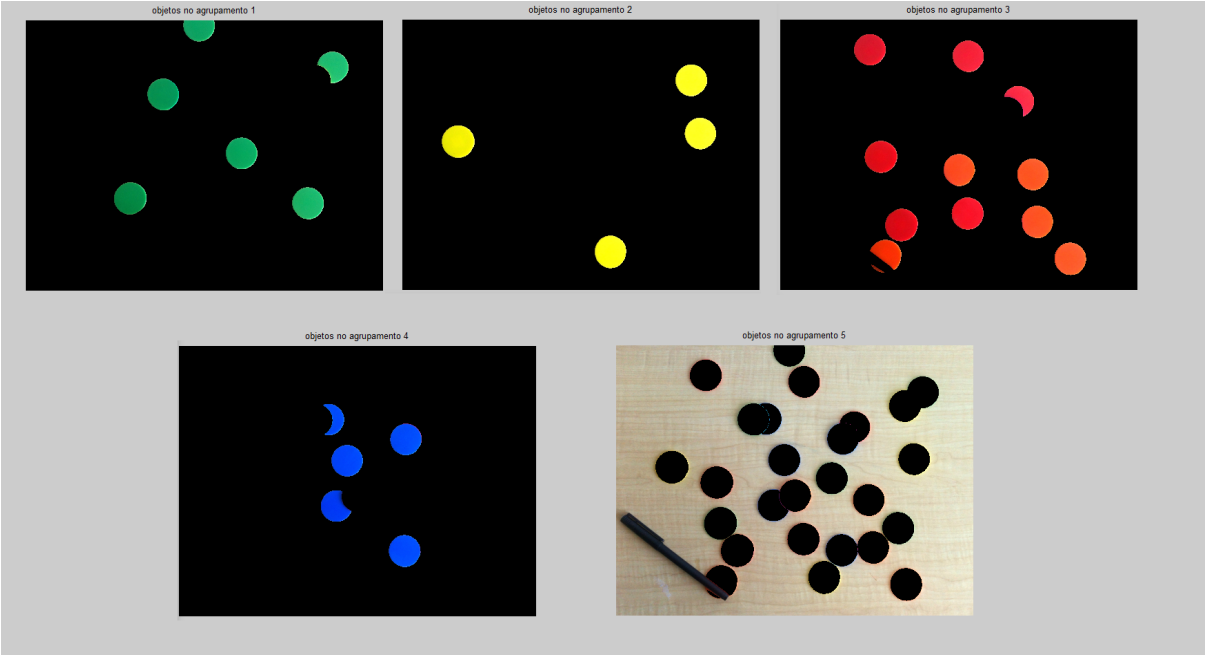
Nosso segundo método de segmentação precisava ser um método com agrupamento, portanto foi escolhido o método de agrupamento por k-médias mostrado em aula, sua implementação foi tranquila porém um aprendizado veio na parte de supervisionar os parâmetros passados para ele.

Inicialmente foi passado um valor de 3 para a quantidade de grupos de cores para ele separar os pixels, isso gerou o seguinte resultado:



Devido à falta de grupos de cores o algoritmo agrupou o amarelo da mesa e das fichas com o grupo de cores verde, portanto aumentamos a quantidade gradualmente até que encontramos o

valor de 5 para isolar todas as fichas de cores visualmente diferentes da mesa, gerando o seguinte resultado:



https://users.cs.cf.ac.uk/Dave.Marshall/Multimedia/node231.html

https://www.math.cuhk.edu.hk/~lmlui/dct.pdf

https://en.wikipedia.org/wiki/Discrete_cosine_transform

https://kb.tourwriter.com/baseline-jpeg-vs-progressive-jpeg

https://medium.com/analytics-vidhya/vector-quantization-using-k-means-algorithm-6382f3888326