MC920 – Introdução ao Processamento Digital de Imagem
Trabalho 5

Rafael Eiki Matheus Imamura - RA 176127[1]

[1]Instituto de Computação – Universidade Estadual de Campinas (Unicamp)
Caixa Postal 6176 – CEP 13083-970 – Campinas – SP – Brasil

ra176127@students.ic.unicamp.br

1. Introdução

Imagens coloridas podem conter uma grande quantidade de informações. Em muitos casos, não sãos necessário todos os detalhes de cores que são guardadas nelas. Para resolver esse problema, pode-se reduzir ou limitar a quantidade de cores que uma imagem pode ter.

Neste trabalho, foi criado um programa para colorir artificialmente uma imagem com uma quantidade limitada de cores. Isso foi feito para imagens JPEG e PNG, usando a técnica de agrupamento *K-means*.

2. Execução

A entrega deste trabalho inclui o presente PDF e um diretório de nome "projeto 5". Nela se encontram o código, as dependências e as imagens de entrada usadas para os testes.

Para executar o código, deve-se ter instalado o OpenCV e o Numpy. Um *freeze* das dependências está presente no arquivo "requirements.txt". Uma alternativa rápida para fazer a instalação é o comando:

pip install -r requirements.txt

Com este comando instalado, é possível chamar o arquivo de script de duas formas.

1. *python main.py example*

   Nesta opção, são gerados os resultados para todas as imagens de exemplo disponíveis no diretório "pictures", com as quantidades de cores: 8, 16, 32, 64 e 128.

2. *python main.py arq_entrada arq_saida cores*

   É gerada uma imagem com a quantidade de cores dada pelo parâmetro *cores*.

   Alguns exemplos de uso são mostrados a seguir:

1. python main.py example
2. python main.py ./pictures/baboon.png ./results/result.png 32

2.1. Entrada

A entrada é dada por 4 imagens PNG disponibilizadas no enunciado do trabalho (http://www.ic.unicamp.br/~helio/imagens_coloridas/).

Todas as entradas usadas estão disponíveis dentro do zip, no diretório “pictures”.

2.2. Saída

A saída do programa são imagens no formato usado para a entrada, com a limitação de cores dada.

3. Solução

A solução adotada usa o método K-means rápido do Scikit-learn. A imagem é transformada do espaço de cores RGB para o LAB, aplicado o método K-means, e a imagem é transformada de volta para o RGB.

4. Resultados

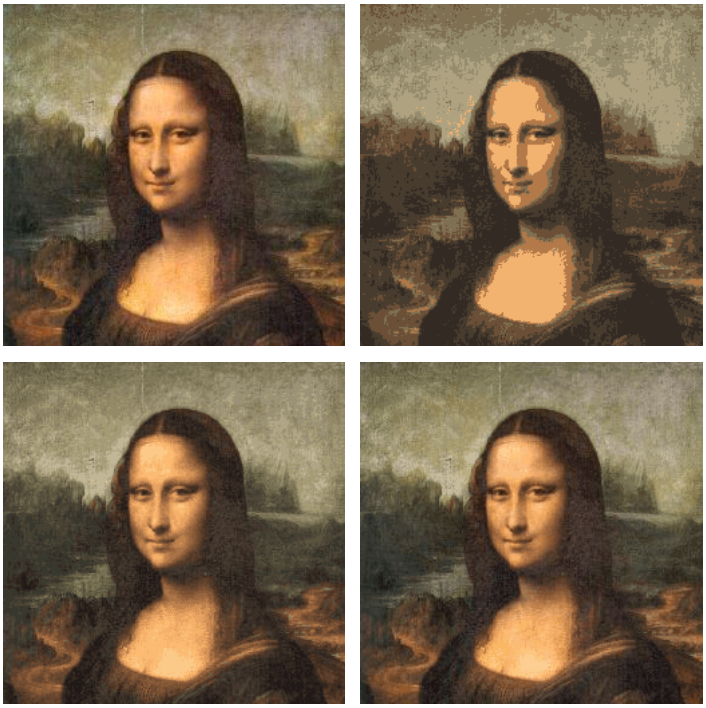

Figura 1. Imagem original “monalisa.png” e versões com 8, 32, e 128 cores.

Nessas imagens da Figura 1 é possível notar a diferença grande na imagem de 8 cores, mas nas demais a diferença visualmente perceptível é mínima. O baixo contraste entre as cores da imagem facilita a boa qualidade usando a técnica. A versão com 8 cores possui tamanho cerca de 3 vezes menor que a original. A de 32 cores é 16% menor que a original. A versão de 128 cores ficou ligeiramente maior que a original.

Figura 2. Imagem original "baboon.png" e versões com 8, 32 e 128 cores.

A Figura 2 apresenta imagens com cores mais distintas. A percepção visual de diferença das imagens, no entanto, é ainda bem pequena. A maior diferença, como era esperado, era na imagem com 8 cores. A versão com 32 cores possui variações maiores no olho esquerdo do macaco. Nas demais regiões, as versões acima de 8 cores se assemelham muito à original. A versão com 8 cores ficou com o tamanho de menos da metade da original, e a de 32 cores ficou com cerca de 25% de redução. A versão de 128 cores é cerca de 8% menor que a original.

5. Conclusões

Com um algoritmo bem simples, usando as bibliotecas fornecidas, foi possível criar um programa que consegue comprimir imagens com a perda de qualidade mínima em muitos casos. Para a criação de versões da imagem que tenham resolução menor, este tipo de solução pode trazer muitos benefícios, já que as perdas são reduzidas. As versões com 128 cores não apresentaram reduções significativas do tamanho das imagens, mas mantiveram uma alta fidelidade em relação à imagem original. Podem haver casos que a imagem original pode ser até menor que a quantidade de cores; para uma compressão mais adequada, é necessário analisar o esquema de cores da imagem.

Referências

SciPy.org. (2019) "NumPy Reference". Disponível em: < https://docs.scipy.org/doc/numpy/reference/>. Acesso em: jun. 2019.