

Trabalho 2

MO443/MC920 - Introdução ao Processamento de Imagem Digital

Lucas de Oliveira Silva - 220715

1 Introdução

Neste trabalho o objetivo foi aplicar a técnica de meios-tons usando difusão de erro. Dada uma imagem e um inteiro k (entre 2 e 256) o algoritmo produz uma saída em que cada banda de cor tem k valores distintos igualmente distribuídos. Por exemplo, para $k=4$, cada banda de cor da imagem de saída terá valores 0, 85, 170 ou 255. A solução foi feita com Python 3.8.5 em um único arquivo .py. Foram utilizadas as seguintes bibliotecas: numpy (1.19.1) e scikit-image (0.16.2).

2 Como executar

A implementação está no *script* trabalho2.py, as imagens de teste na raiz e as saídas obtidas na pasta output. O programa precisa necessariamente de uma imagem de entrada (-i img) em formato png e um inteiro (-num n), entre 1 e 6, que indica a máscara que será usada conforme a ordem no enunciado. O resto dos parâmetros são opcionais e possuem valores padrões. Os parâmetros opcionais são os seguintes:

- -o Indica a imagem de saída. O padrão é out.png.
- -zig Indica se a imagem deve ser percorrida pela varredura em zigue-zague. O padrão é falso.
- -k Quantos níveis usar em cada banda de cor na imagem de saída. O padrão é 2, ou seja, os valores são 0 ou 255.

Ex: python3 trabalho2.py -i ./watch.png -o ./saida.png -num 1 -zig true

Para mais detalhes é possível executar o *script* com o parâmetro -h.

A imagem é lida usando a função `io.imread` do skimage, e logo depois os valores são convertidos para float. Logo depois a máscara indicada por parâmetro é usada pelo algoritmo. O tratamento de bordas é feito redimensionando a imagem e preenchendo os espaços vazios com zero. Logo parte do erro é perdido próximo às bordas, mas isso não altera em nada a percepção visual do resultado.

Antes de salvar a imagem em arquivo é feito um arredondamento seguido de um *clipping* para o intervalo [0, 255].

3 Avaliação dos resultados

Os resultados obtidos estão na pasta output, mas para ilustrar as figuras de 1 a 4 são mostradas nesse trabalho. Em geral as máscaras produziram saídas similares. Os artefatos foram levemente reduzidos ao varrer a imagem em zig-zague. Mesmo com as saídas parecidas a olho nú, a imagem da máscara (b) de Stevenson e Arce ficou levemente mais escura e destacou as bordas originais (mais notável nos pelos do focinho). Ao variar o parâmetro k foi possível ver a imagem ganhando cor novamente, mas a partir de $k=16$ a diferença é imperceptível.

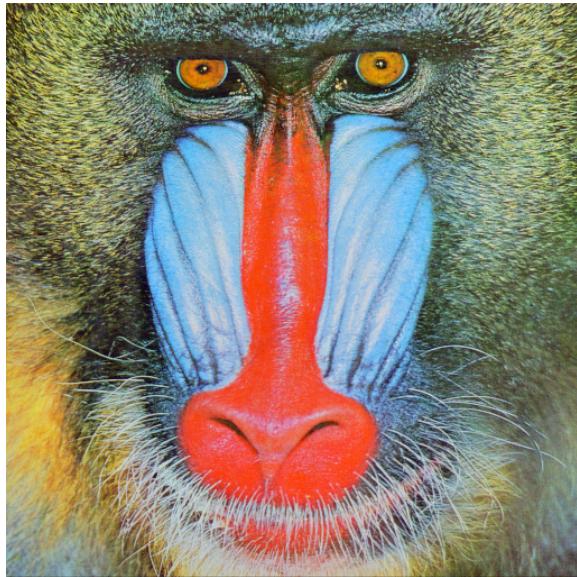
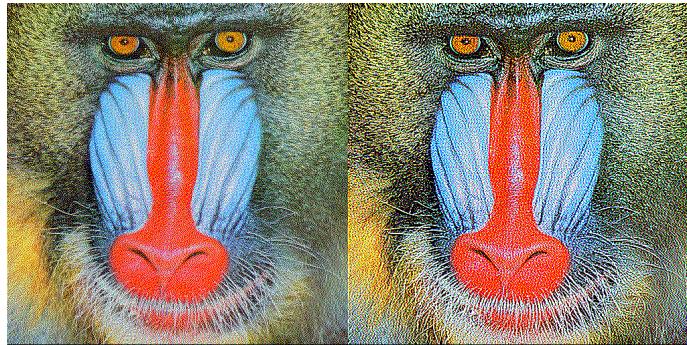
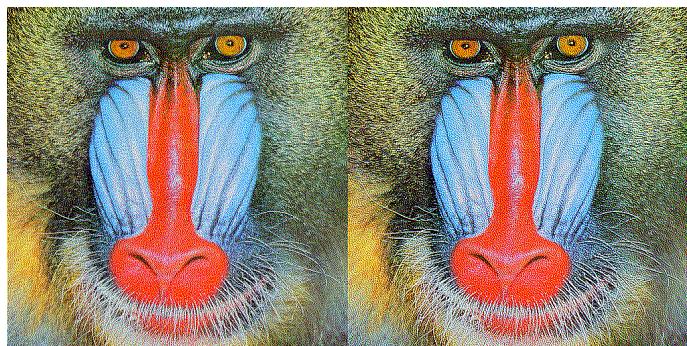


Figura 1: Imagem baboon.png original



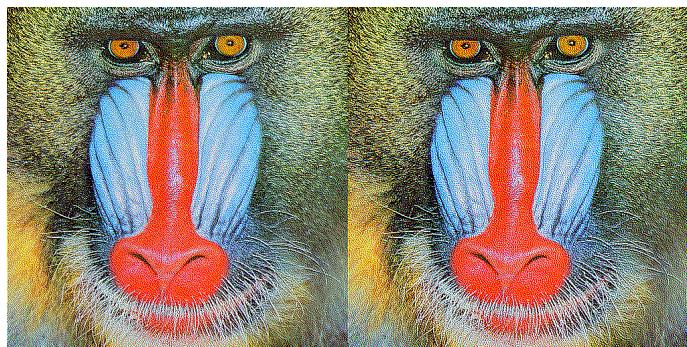
(a) Floyd e Steinberg

(b) Stevenson e Arce



(c) Burkes

(d) Sierra



(e) Stucki

(f) Jarvis, Judice e Ninke

Figura 2: Resultado da aplicação de cada máscara a imagem baboon.png com varredura linha a linha

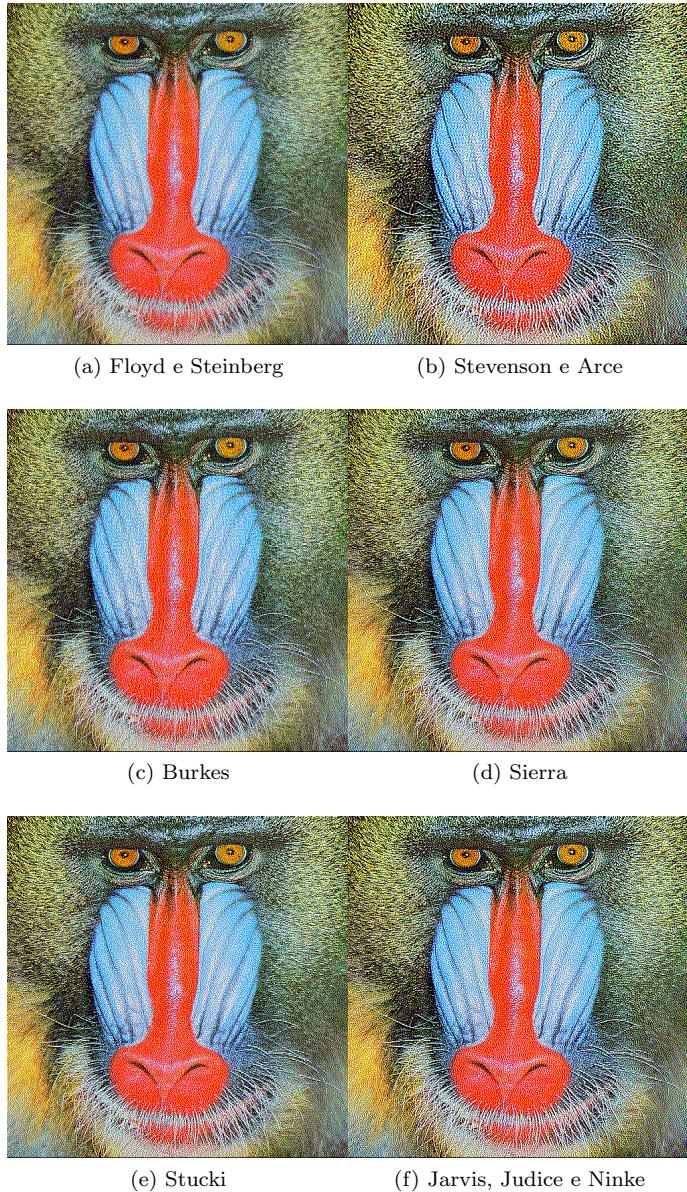


Figura 3: Resultado da aplicação de cada máscara a imagem baboon.png com varredura zigue-zague

4 Conclusão

Os devidos arredondamentos e tratamentos foram aplicados e portanto não há limitações quanto a valores na imagem de entrada/saída.

Em relação ao resultado, o efeito foi muito interessante já que mesmo com $k=2$ (menor valor) as saídas ainda possuem uma boa percepção visual, isso usando apenas 8 cores distintas. Para k menor que 16 a saída possui alguns artefatos que se destacam ao ampliar a visualização. Mas com k maior ou igual a 16 a diferença da imagem real já é quase imperceptível, portanto é possível comprimir a imagem e usar apenas 4 bits para cada banda de cor ao invés de 8.

O *script* final demora bastante pra executar, principalmente com imagens grandes. Mas não foi possível vetorizar todo o algoritmo dada a dependência de cada iteração aos valores anteriores. Portanto algumas otimizações foram feitas apenas no laço mais interno. Já que cada banda de cor é tratada de forma independente uma possível melhoria seria paralelizar a aplicação do algoritmo e aproveitar mais a CPU.

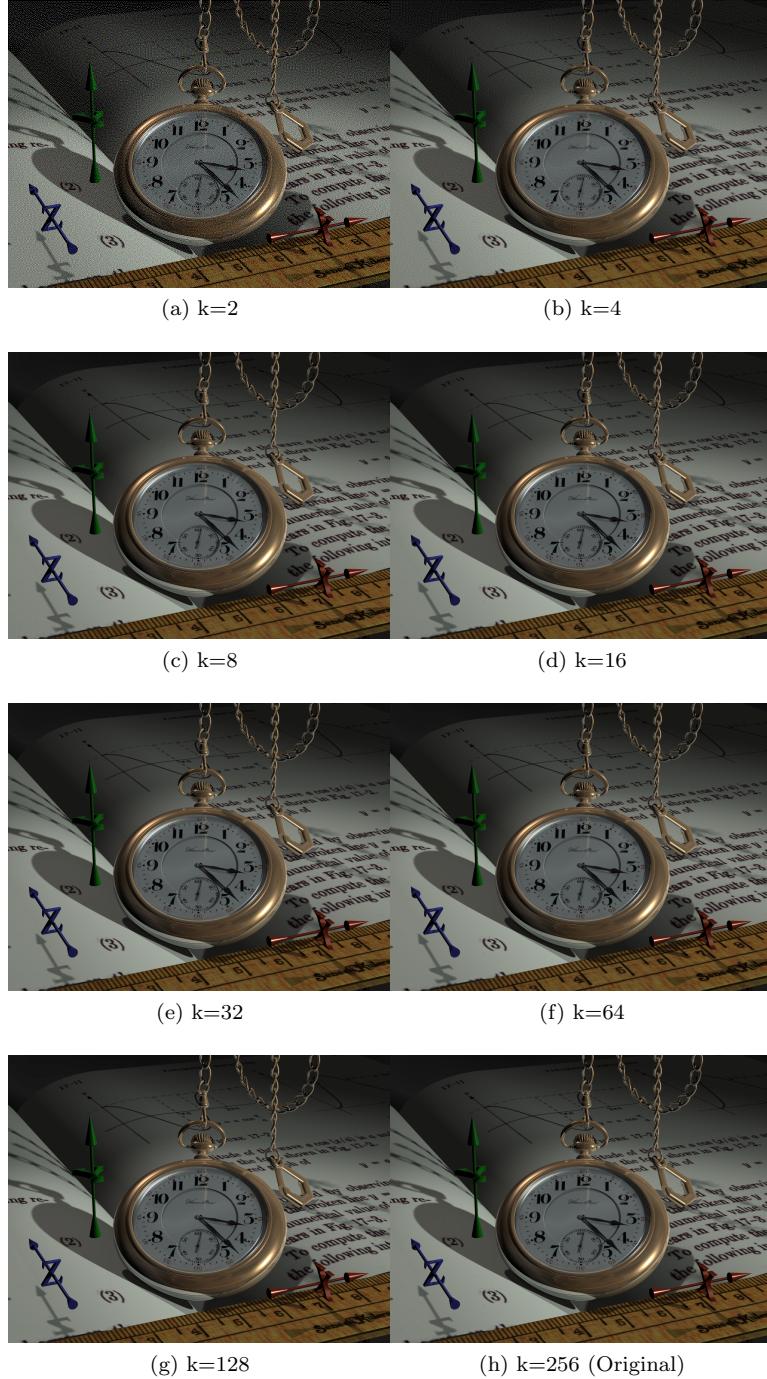


Figura 4: Resultado da aplicação da máscara (a) de Floyd e Steinberg a imagem watch.png variando o parâmetro k