Trabalho 2 - Introdução ao Processamento de Imagem Digital

Enoque Alves de Castro Neto (230211)[1]

[1]Universidade Estadual de Campinas
Instituto de Computação

enoquealvesufc@gmail.com

1. Introdução

Este relatório foi desenvolvido para detalhar o funcionamento e os resultados do trabalho 2 da disciplina *MO443*. O objetivo do trabalho é implementar algumas técnicas de pontilhado, utilizadas para exibir uma imagem, procurando manter uma boa percepção por parte do usuário. Os algoritmos implementados foram o algoritmo de pontilhado ordenado e o algoritmo de pontilhado por difusão de erros.

Toda a implementação do trabalho foi feita no jupter-notebook e cada função está separada por bloco. Então basta executar os blocos separadamente para usar cada função.

2. Implementação

2.1. Visão Geral

Para a implementação deste trabalho foi utilizado a linguagem Python 3.7.3, com o uso das bibliotecas OpenCV, Numpy, Matplotlib e imageio. De maneira geral, a biblioteca Numpy foi utilizada para a manipulação das imagens através de arrays Numpy. A biblioteca OpenCV foi usada apenas para carregar a imagem e armazenar em um array Numpy. Já a biblioteca Matplotlib foi usada para plotar as imagens geradas em cada requisito para este trabalho e o imagemio foi utilizado para salvar as imagens no formato *.pbm*. Foi notado durante a implementação deste trabalho uma grande diferencia entre as imagens resultados por parte do Matplotlib e o imagemio.

2.2. Carregamento da imagem

Para o carregamento da imagem, foi utilizada a função *cv2.imread("diretório_da_imagem.png", cv2.IMREAD_GRAYSCALE)* do OpenCV que retorna sua matriz de pixel. O primeiro argumento da função indica o diretório da imagem e o segundo indica que estamos interessados apenas nas escalas de cinzas. Para o segundo argumentos também temos as seguintes opções:

1. cv2.IMREAD_COLOR: Carrega uma imagem colorida. Qualquer transparência dessa imagem será negligenciada. Caso nenhum argumento seja colocado, essa opção será a padrão.
2. cv2.IMREAD_UNCHANGED: Carrega uma imagem tal como ela é, incluindo um canal alpha.

2.3. Pontilhado por ordenado

O algoritmo de pontilhado ordenado utiliza um conjunto de padrões formados por pontos pretos e brancos. Dois conjuntos de padrões foi utilizado para os experimentos realizados e eles podem ser observados na figura 1 e 2.

$$M_{3\times3} = \begin{array}{|c|c|c|}\hline 6 & 8 & 4 \\ \hline 1 & 0 & 3 \\ \hline 5 & 2 & 7 \\ \hline \end{array}$$

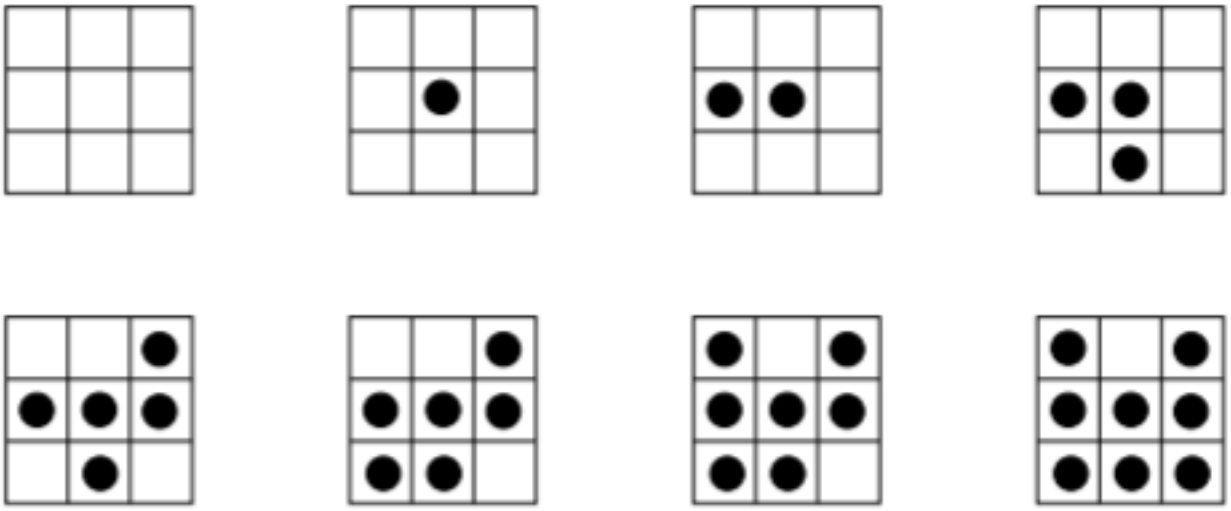

Figura 1. Padrão gerado pela a matriz $M_{3\times3}$

$$M_{4\times4} = \begin{array}{|c|c|c|c|}\hline 0 & 12 & 3 & 15 \\ \hline 8 & 4 & 11 & 7 \\ \hline 2 & 14 & 1 & 13 \\ \hline 10 & 6 & 9 & 5 \\ \hline \end{array}$$

Figura 2. Segunda matriz utilizada para a criação dos padrões

Para a implementação do algoritmo pontilhado foi criada uma função chamada *cria_padroes(matriz_padrao)* que recebe uma matriz representando o padrão desejado e retorna um array de matriz de padrões. Para a matriz de padrões da imagem 1, o retorno desta função será um array de 10 posições, tal que cada posição do array possui uma matriz 3×3 das que estão presente na imagem.

Após criada a função *cria_padroes*, foi criada a função *half_toning(matriz_de_padroes)* que normaliza a imagem para possuir somente valores de [0,$tam\_imagem^2 - 1$]. Após normalizar, uma nova imagem de $altura * tam\_padrao \times largura * tam\_padrao$ é gerado e é adicionado o padrão

correspondente ao valor normalizado da imagem original a cada pixel original da imagem, por esse motivo o tamanho da imagem é aumentando de acordo com o tamanho do padrão.

2.4. Pontilhado por difusão de erro

O algoritmo de pontilhado por difusão de erro também transforma a imagem original em uma imagem contendo apenas os valores preto e branco, entretanto, leva em considera¸c˜ao os valores ao redor de cada pixel. A técnica de pontilhado por difusão de erro de Floyd-Steinberg é resumida a seguir:

1. percorra todos os pixels da imagem, segundo uma ordem pré-definida.
2. para cada pixel, se seu valor for maior do que 128, troque-o para 255 (branco). Caso contrário, troque-o para 0 (preto). Armazene o erro, ou seja, a diferença entre o valor exato do pixel e o valor aproximado.
3. distribua o erro aos pixels adjacentes, da seguinte forma (figura 3):
    a) adicione 7/16 do erro ao pixel localizado à direita.
    b) adicione 3/16 do erro ao pixel localizado abaixo e à esquerda.
    c) adicione 5/16 do erro ao pixel localizado abaixo.
    d) adicione 1/16 do erro ao pixel localizado abaixo e à direita.

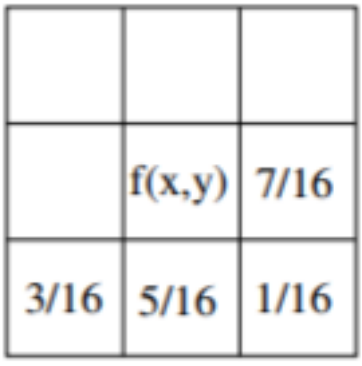

Figura 3. Distribuição de erro na técnica de pontilhado com difusão de erro de Floyd-Steinberg.

A ordem na qual a imagem é percorrida pode produzir resultados diferentes no processo de dithering. A varredura da esquerda para a direita (figura 4(a)) pode gerar padrões indesejados ou a impressão de uma certa direcionalidade na imagem resultante. Para evitar esses efeitos, uma alternativa é modificar a direção de varredura a cada linha (figura 4(b)).

Figura 4. Formas de varredura da imagem.

Para este trabalho foi implementado as duas formas de varredura da imagem. Para a varredura apresentada na figura 4(b), é necessário tomar um cuidado em especial, pois em algumas interações o erro é propagado para esquerda/baixo e em outras interações o erro é propagado para direita/baixo.

3. Resultados

Foram escolhidas duas imagens para os experimentos deste trabalho: *'baboon.pgm'* e *'monarch.pgm'*. O arquivo de saida gerado tem formato *'pbm'*.

3.1. Pontilhado ordenado

Para o padrão apresentado na imagem 1, o resultado para imagem *'monarch'* é o seguinte:

Figura 5. Imagem monarch original

Figura 6. Imagem monarch após a aplicação do pontilhado ordenado

Pode se notar que visualmente é quase impossível dizer apenas olhando para imagem que existem apenas pixeis com cores braco ou preto. A imagem original, que era de tons de cinza, se tornou uma imagem sem perdas de muitos detalhes sendo agora preto ou branca. O grande problema dessa técnica de pontilhado ordenado é que a imagem resultante crescerá em relação ao tamanho do padrão, por exemplo, se o padrão for tamanho 3×3, a imagem resultante será 9 vezes maior. Para a imagem do baboon, o resultado foi até mais satisfatório que na imagem do monarch, tal fato talvez tenha acontecido por

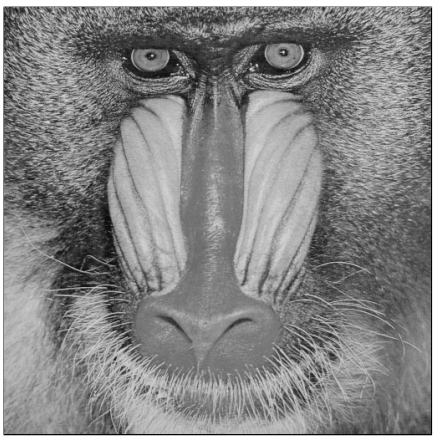

Figura 7. Imagem baboon original

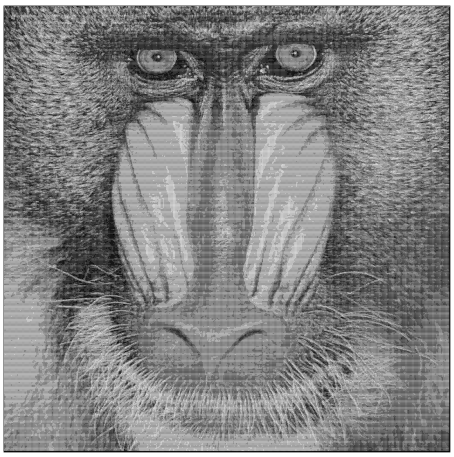

Figura 8. Imagem baboon após a aplicação do pontilhado ordenado

conta da imagem baboon tenha menos detalhes de paisagens e coisa do tipo. O resultado do baboon pode ser observado na figura 8.

Também foi utilizado o padrão apresentado na figura 2, e os resultados foram os seguintes:

Figura 9. Imagem monarch original

Figura 10. Imagem monarch após a aplicação do pontilhado ordenado(b)

É nítido o ganho de detalhes da imagem monarch apresentada na imagem 10. O problema de falta de alguns detalhes do resultado anterior foi praticamente zerado e a imagem resultante é praticamente igual a original para o olho humano. Porém, como comentado anteriormente, o grande problema do método pontilhado ordenado é que o tamanho da imagem resultante crescerá em relação do tamanho do padrão. Como o padrão apresentado na figura 2 possui tamanho 4×4, a imagem resultante terá tamanho 16 vezes maior que a original. Tendo isso em mente, dependendo da aplicação, utilizar esse método com esse padrão pode ser inviável em alguns casos, pois a imagem resultante será muito grande. Para a imagem do baboon, o resultado obtido anteriormente já era bastante satisfatório, então não vale a pena utilizar esse padrão para essa imagem, tendo em vista que aumentará bastante o tamanho da imagem.

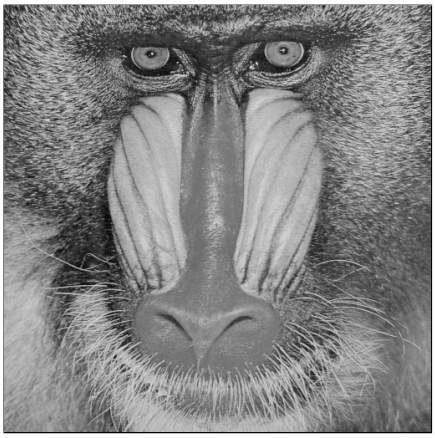

Figura 11. Imagem baboon original

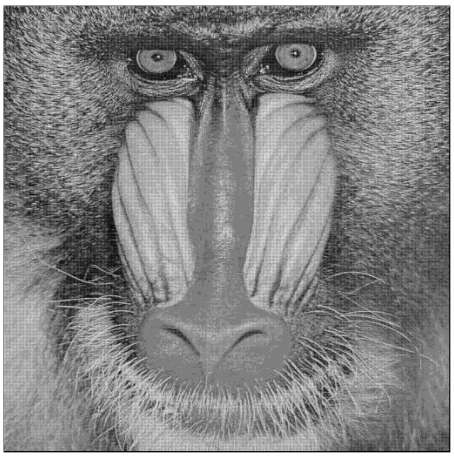

Figura 12. Imagem baboon após a aplicação do pontilhado ordenado(b)

3.2. Pontilhado por difusão de erro

Para a técnica de pontilhado por difusão de erro foi utilizado as duas formas de varreduras apresentado na figura 4. Os resultados obtidos podem ser visto a seguir:

Figura 13. Imagem monarch original

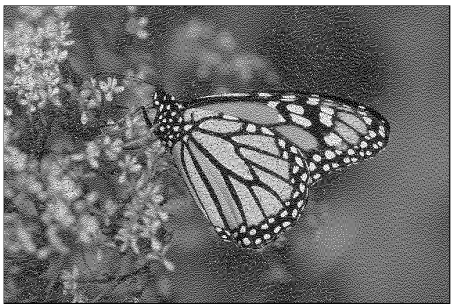

Figura 14. Imagem monarch após a aplicação do pontilhado por difusão de erro(a)

O resultado para a imagem *monarch* apresentada na figura 14 parece até bem satisfatório. Para a figura 14, o método de varredura utilizado foi o apresentado na figura 4(b). O que ocorre negativamente é um pouco de mudança na textura da imagem, principalmente nos detalhes do fundo da imagem. Em compensação a isso, o resultado desse método é imagem com o tamanho igual ao tamanho da imagem original, não precisando um tamanho extra para sua utilização. Já para imagem do *baboon*, o resultado novamente praticamente igual ao original, se beneficiando novamente da falta de detalhes da imagem original.

O resultado obtido para a forma de varredura apresentada na figura 4(b) será apresentado a seguir:

O resultado para imagem *monarch* utilizando a varredura apresentada na figura 4(a) pode ser visualizada na figura 18. A utilização do método da figura 4(a) e 4(b) foram praticamente iguais, tendo como principal diferença apenas alguns poucos detalhes nas imagens que foram perdidos na imagem 18 em relação a imagem 14, porém essa diferença

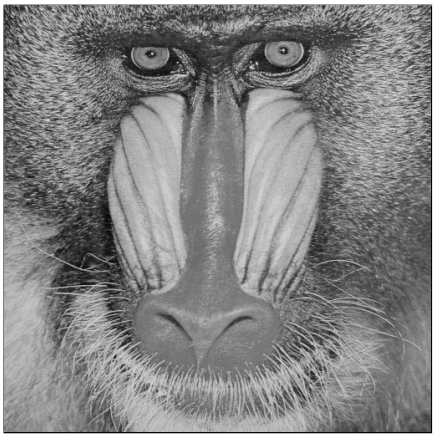

Figura 15. Imagem baboon original

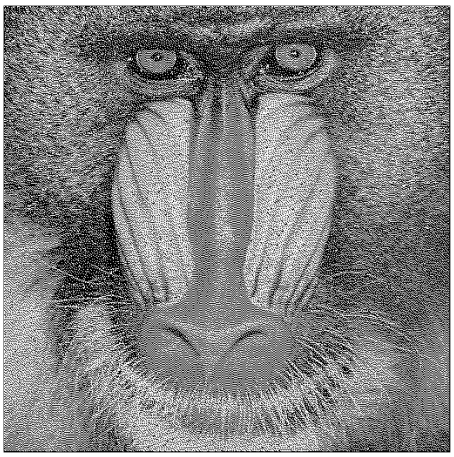

Figura 16. Imagem baboon após a aplicação do pontilhado por difusão de erro(a)

Figura 17. Imagem monarch original

Figura 18. Imagem monarch após a aplicação do pontilhado por difusão de erro(b)

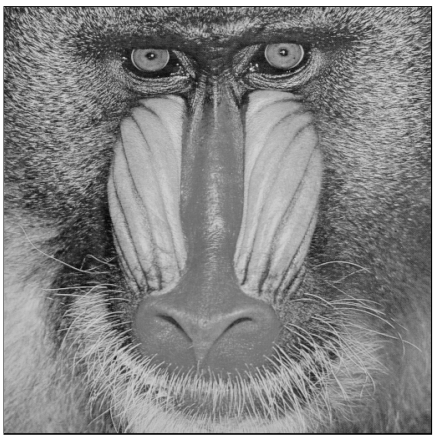

Figura 19. Imagem baboon original

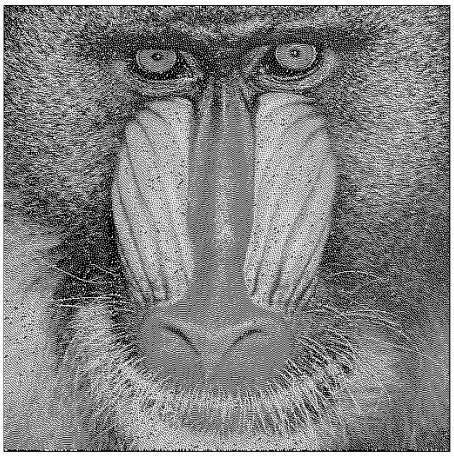

Figura 20. Imagem baboon após a aplicação do pontilhado por difusão de erro(b)

quase não é notado ao olho humano. Novamente, para a figura do *baboon* o resultado foi praticamente igual ao original, como na figura 16.

4. Conclusão

Neste trabalho foi utilizadas as técnicas de pontilhado para reduzir a quantidade de cores usadas para exibir uma imagem, procurando manter uma boa percepção por parte do usuário. Em todos os resultados obtidos é incrível a fidelidade do resultado com a original, levando em conta que a imagem gerada possui somente duas cores diferentes e a imagem original possui 256 cores diferentes.

Referências

Pedrini, H. and Schwartz, W. R. (2008). *Análise de imagens digitais: princípios, algoritmos e aplicações*. Thomson Learning.