Trabalho MC920 2:
Técnicas de Pontilhado

Esdras R. Carmo - RA 170656

8 de maio de 2019

1 Introdução

Exploraremos nesse trabalho a técnica de pontilhado em imagens monocromáticas, que visa reduzir os níveis de cinza da imagem, isto é, sua profundidade.

Estamos interessados em estudar algoritmos que recebem uma imagem em escala de cinza e retorna uma imagem com pixels pretos e brancos de modo a manter boa percepção por parte do usuário. Os algoritmos estudados serão duas técnicas diferentes de pontilhados ordenados e o algoritmo de pontilhado por difusão de erro de Floyd-Steinberg.

2 Especificação do Problema

O algoritmo de pontilhado ordenado consiste em substituir cada pixel da imagem original por um conjunto de pixels pretos e brancos seguindo um padrão de pontos pretos e brancos. Por exemplo, se utilizarmos um conjunto de pixels 3×3 como definido em (2) conseguimos 10 padrões diferentes (ver Figura 1) variando entre todos os pixels apagados e todos os pixels acesos.

$$M_{3\times3} = \begin{bmatrix} 6 & 8 & 4 \\ 1 & 0 & 3 \\ 5 & 2 & 7 \end{bmatrix} \tag{1}$$

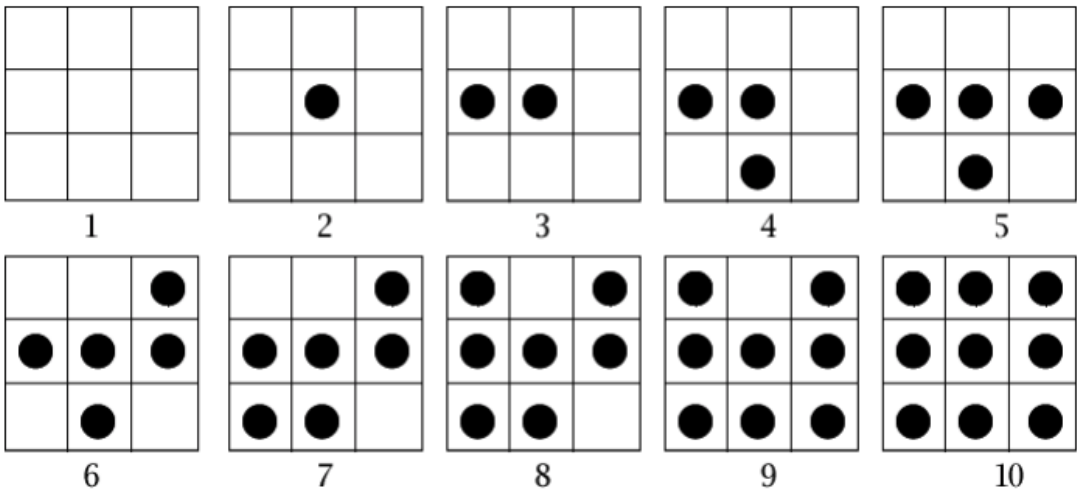


Figura 1: Padrões para a $M_{3\times3}$

Outra máscara que também será utilizada para o pontilhado ordenado é a máscara de Bayer:

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

O algoritmo de pontilhado por difusão de erro de Floyd-Steinberg consiste em percorrer todos os pixels da imagem em uma ordem, aplicar um limiar para substituir o valor do pixel para 0 ou 255, calcular o erro causado por esse limiar e propagá-lo para os pixels vizinhos ainda não visitados.

Caso o percurso seja da esquerda para a direita e de cima para baixo, os pesos para propagar os erros serão:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 7/16 \\ 3/16 & 5/16 & 1/16 \end{bmatrix}$$

Caso o percurso seja da direita para esquerda e de cima para baixo, a matriz deverá ser espelhada:

$$\begin{bmatrix} 0 & 0 & 0 \\ 7/16 & 0 & 0 \\ 1/16 & 5/16 & 3/16 \end{bmatrix}$$

3 Entrada de Dados

O script para aplicação dos algoritmos foi escrito utilizando Python 3.7 com as bibliotecas *numpy* e *opencv*. Para a execução do código basta rodar o seguinte comando:

```
python script.py input_image.pgm output_image.pgm
```

Sendo os parâmetros:

- *script.py*: Nome do script a ser utilizado, escolhido entre *half_ toning.py* e *floyd_ steinberg.py*;
- *input_ image.pgm*: Caminho da imagem de entrada, aceitando imagens em pgm em escala de cinza;
- *output_ image.pgm*: Caminho da imagem de saída, no formato pgm em escala de cinza;

4 Detalhes de Implementação

Para a leitura e escrita da imagem em tons de cinza foi utilizado as funções *imread* e *imwrite* do OpenCV, que suporta também o formato pgm de imagem.

4.1 Pontilhados Ordenados

Para a técnica de pontilhados ordenados, primeiramente a matriz da imagem de entrada é quantizada em valores inteiros no intervalo $[0; 9]$ para a máscara $M_{3\times3}$ e o intervalo $[0; 16]$ para a máscara $M_{4\times4}$.

Feito isso uma nova matriz que representará a imagem de saída pontilhada é criada replicando a máscara escolhida w vezes na horizontal e h vezes na vertical, sendo w e h a largura e altura da imagem de entrada respectivamente. Dessa forma, a imagem de saída terá dimensões iguais ao produto das dimensões da imagem de entrada pela dimensão da máscara.

Por exemplo, para $w = 2$ e $h = 3$, tomando como máscara a matriz M:

$$M = \begin{bmatrix} 1 & 0 \\ 3 & 2 \end{bmatrix}$$

teremos ao fim desse processo a matriz de saída M_o:

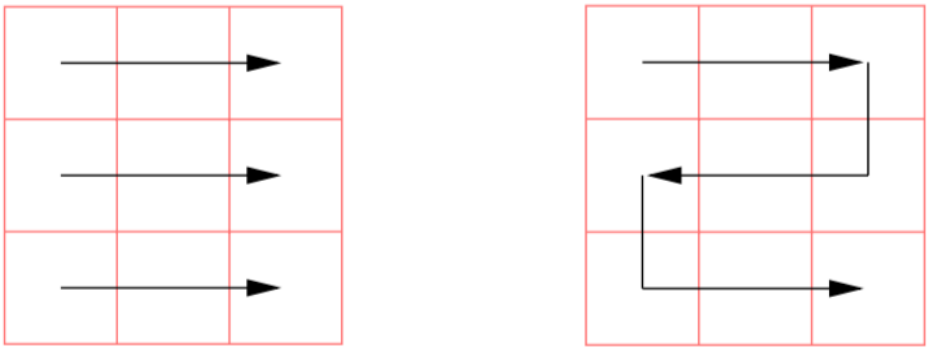
$$M_o = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 3 & 2 & 3 & 2 \\ 1 & 0 & 1 & 0 \\ 3 & 2 & 3 & 2 \\ 1 & 0 & 1 & 0 \\ 3 & 2 & 3 & 2 \end{bmatrix}$$

Por fim realiza-se o processo de pontilhamento, que consiste em percorrer e atualizar a imagem de saída M_o pixel a pixel aplicando o limiar através da seguinte relação com a imagem de entrada M_i, sendo d a dimensão da matriz M utilizada como máscara:

$$M_o(i,j) := \begin{cases} 255, & \text{se } M_o(i,j) < M_i(\lfloor i/d \rfloor, \lfloor j/d \rfloor) \\ 0, & \text{se } M_o(i,j) >= M_i(\lfloor i/d \rfloor, \lfloor j/d \rfloor) \end{cases}$$

4.2 Pontilhados com Difusão de Erro

Para o pontilhado com difusão de erro de Floyd-Steinberg foi decidido entre dois padrões de varredura na imagem, da esquerda para a direita (figura 2a) e em zig-zag (figura 2b)



(a) (b)

Figura 2: Padrão de varredura no pontilhado com difusão de erro

Percebemos que em regiões homogêneas a varredura em zig-zag teve melhor performance em não criar padrões indesejados, i. e., linhas direcionadas de pixels como pode ser observado na figura 3.

Dessa forma, para todos os resultados com o algoritmo de pontilhado com difusão de erro foi utilizado a varredura em zig-zag.

5 Resultado e Discussão

Os resultados comparando os algoritmos utilizados estão apresentados nas figuras 4 e 5.

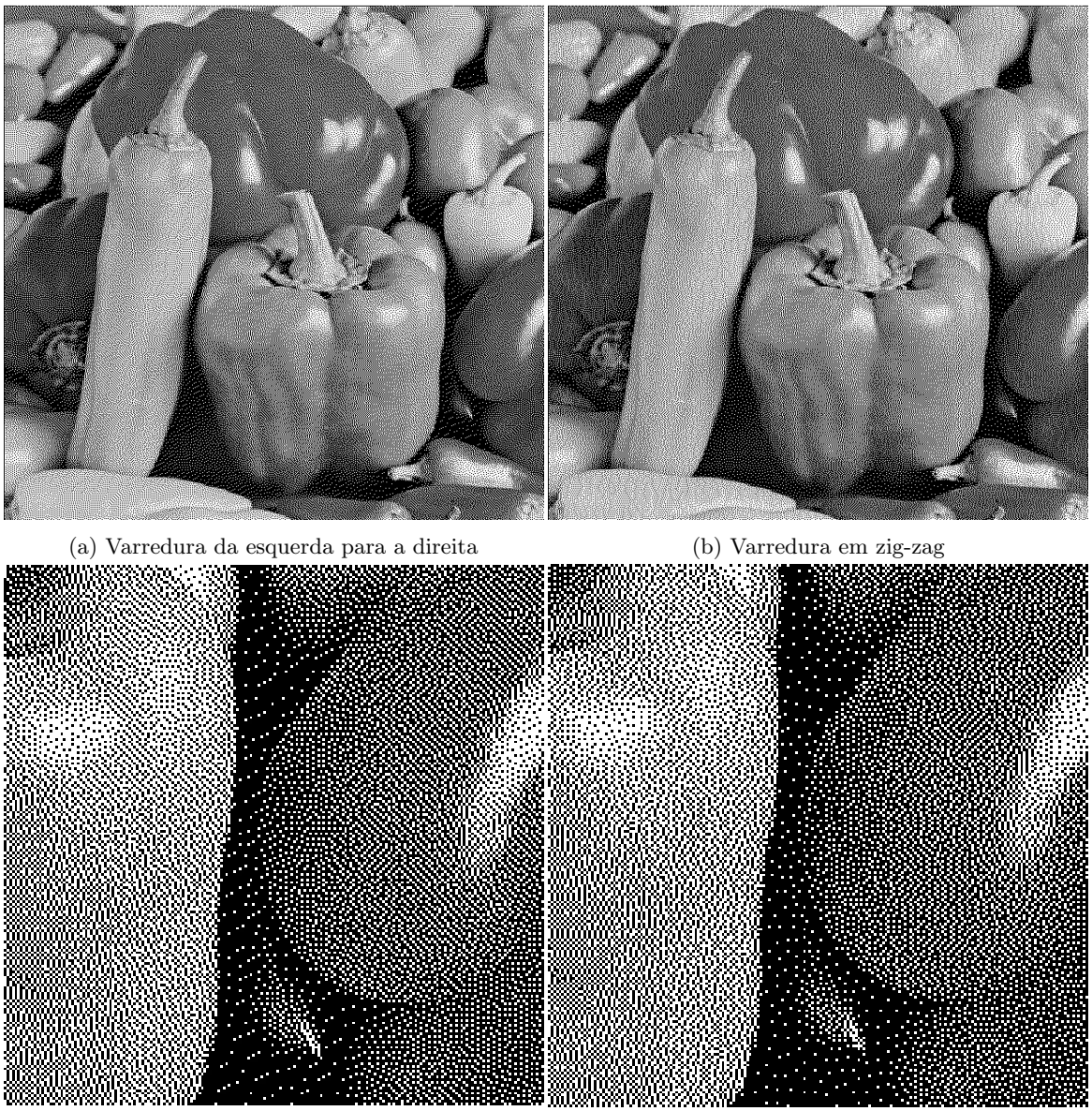
Percebe-se que os pontilhados ordenados realçam detalhes presentes na imagem original, enquanto o pontilhado com difusão de erro de Floyd-Steinberg suaviza os mesmos. No entanto, devido a maior dimensão das imagens de saída do pontilhado ordenado, observa-se maior definição em comparação com a difusão de erro, principalmente utilizando a máscara $M_{4\times4}$.

O pontilhado com difusão de erro mantém as dimensões da imagem de entrada, sendo assim uma vantagem em relação ao pontilhado ordenado que aumenta a imagem de tamanho consideravelmente.

6 Conclusão

Nesse trabalho exploramos e comparamos duas técnicas de pontilhados. Verificamos que é possível transformar uma imagem em tons de cinzas em apenas pontos pretos e brancos mantendo uma boa quantidade de detalhes, e até mesmo sem aumentar a dimensão da imagem caso utilize o algoritmo de Floyd-Steinberg.

Dentre os algoritmos implementados, o que obteve melhor resposta em manter os detalhes da imagem foi o pontilhado ordenado com máscara $M_{4\times4}$, enquanto o Floyd-Steinberg se mostrou bem efetivo sem necessitar aumentar o tamanho da imagem.



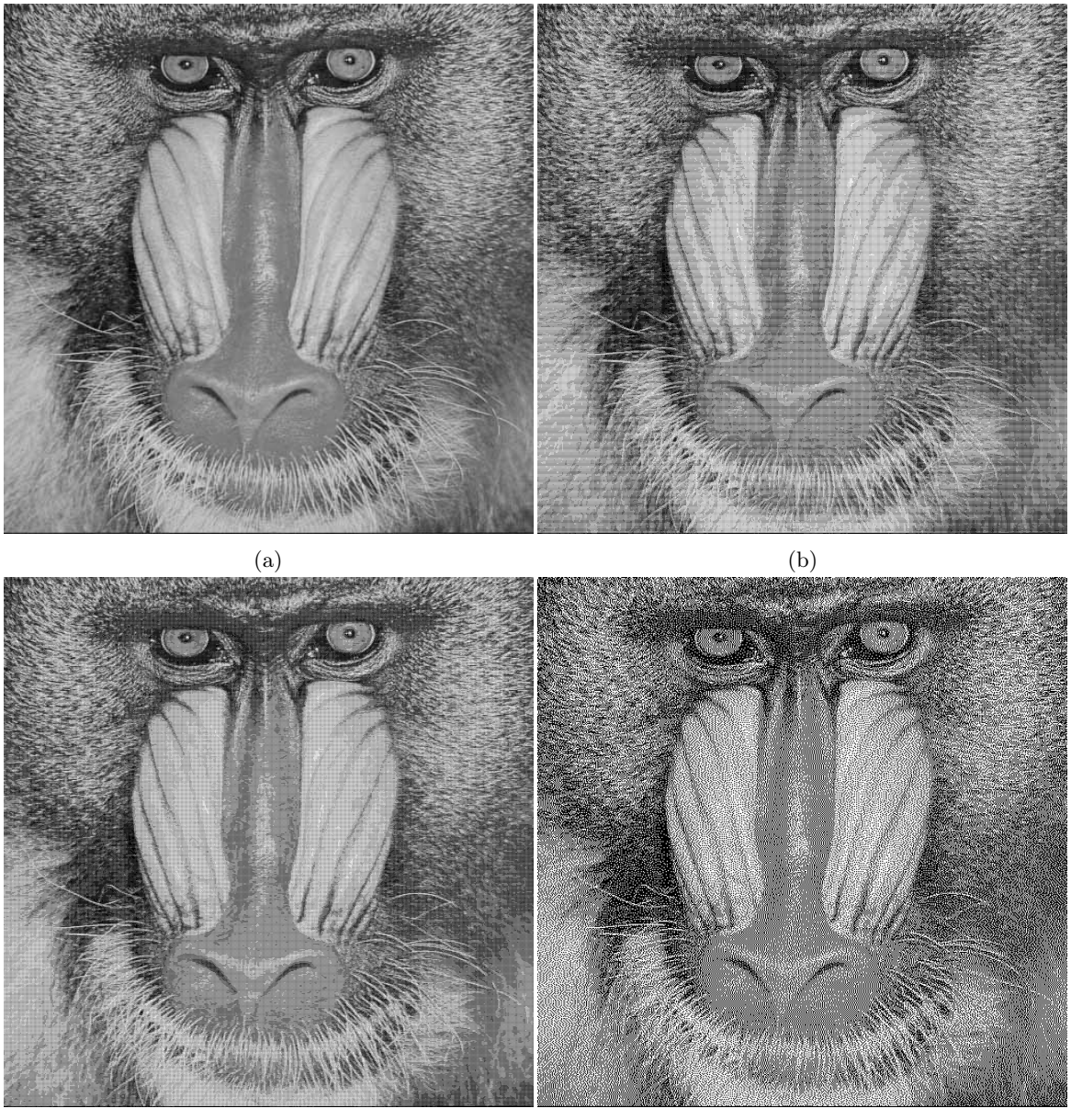
(a) Varredura da esquerda para a direita

(b) Varredura em zig-zag

(c) Varredura da esquerda para a direita

(d) Varredura em zig-zag

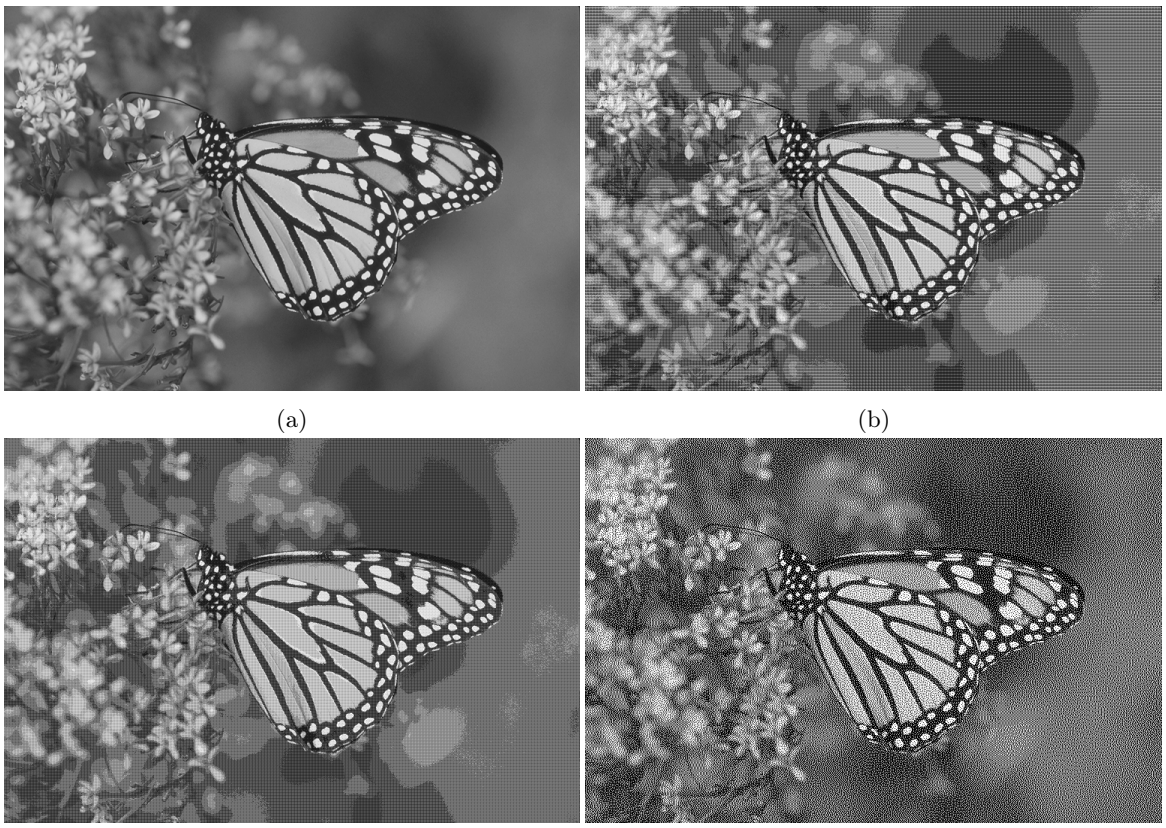
Figura 3: Comparação entre padrões de varredura



(a) (b)

(c) (d)

Figura 4: Comparação entre pontilhados: (a) Imagem original (b) Pontilhado ordenado com máscara $M_{3\times3}$ (c) Pontilhado ordenado com máscara $M_{4\times4}$ (d) Pontilhado com difusão de erro



(a) (b)

(c) (d)

Figura 5: Comparação entre pontilhados: (a) Imagem original (b) Pontilhado ordenado com máscara $M_{3\times3}$ (c) Pontilhado ordenado com máscara $M_{4\times4}$ (d) Pontilhado com difusão de erro