

Trabalho 1

Tiago Loureiro Chaves (187690)

MC920A - *Introdução ao Processamento de Imagem Digital* - 2s2019

Resumo

Este projeto teve como objetivo avaliar diferentes técnicas de meios-tons com difusão de erro (*error diffusion dithering*) para redução da quantização de cores de imagens.

Além da forma clássica de Floyd-Steinberg, cinco outros métodos foram implementados, tanto para imagens coloridas como para imagens monocromáticas.

1 Problema

Técnicas de meios-tons (*halftoning* [1]) são empregadas quando queremos diminuir a quantidade de cores utilizadas para a representação de imagens, no caso deste projeto, para apenas duas cores em cada canal (0 ou 255).

Uma abordagem ingênuas seria a limiarização da imagem [2], porém, como vemos na Figura 1b, gradientes na imagem são praticamente perdidos ao serem substituídos por uma cor uniforme, podendo tornar a imagem irreconhecível em alguns casos.



(a) Imagem original

(b) Limiarização

(c) Floyd-Steinberg

Figura 1: Comparação de técnicas de meios-tons (com *threshold* de 128)

Dessa forma, surgem as técnicas de *dithering* [3] com o objetivo de combinar as cores disponíveis de maneira a melhor imitar as indisponíveis, baseando-se no fato de nossos olhos mesclarem pequenos pontos de cores distintas, registrando uma intensidade e matiz geral.

Em particular, exploraremos as técnicas com difusão de erro, que resultam em menos artefatos visuais do que as de *dithering* ordenado [4], ao considerarem o erro "propagado" pela diferença entre a cor definida para pixels adjacentes e seu valor na imagem original. Seis técnicas foram implementadas, cujas distribuições de erro são mostradas na Figura 2:

- 0. Floyd e Steinberg
- 1. Stevenson e Arce
- 2. Burkes
- 3. Sierra
- 4. Stucki
- 5. Jarvis, Judice e Ninke

	$f(x, y)$	7/16
3/16	5/16	1/16

(a) Floyd e Steinberg

			$f(x, y)$		32/200	
12/200		26/200		30/200		16/200
	12/200		26/200		12/200	
5/200		12/200		12/200		5/200

(b) Stevenson e Arce

		$f(x, y)$	8/32	4/32
2/32	4/32	8/32	4/32	2/32

(c) Burkes

		$f(x, y)$	5/32	3/32
2/32	4/32	5/32	4/32	2/32
	2/32	3/32	2/32	

(d) Sierra

		$f(x, y)$	8/42	4/42
2/42	4/42	8/42	4/42	2/42
1/42	2/42	4/42	2/42	1/42

(e) Stucki

		$f(x, y)$	7/48	5/48
3/48	5/48	7/48	5/48	3/48
1/48	3/48	5/48	3/48	1/48

(f) Jarvis, Judice e Ninke

Figura 2: Matrizes de distribuição de erro para diferentes técnicas de *dithering*

2 Programa

2.1 Dependências e Organização

O trabalho foi desenvolvido em Python 3.7.4, com as bibliotecas Numpy 1.17.0, Matplotlib 3.1.1 e OpenCV 4.1.0, e é dividido em três arquivos:

- `main.py` – tratamento da entrada/saída e execução do projeto
- `dithering.py` – implementação das técnicas de meios-tonos
- `image_processing_utils.py` – funções gerais para processamento de imagens

As imagens utilizadas para teste estão na pasta `i/`, e alguns exemplos de saída encontram-se na pasta `o/`. O código também pode ser visto em github.com/laurelkeys/image-processing.

2.2 Execução

As imagens de entrada `.png` devem ser colocadas na pasta `i/`, localizada no mesmo diretório do script `main.py`, que salvará as imagens resultantes na pasta `o/` (também no formato `.png`).

Por padrão, aplicam-se todas as técnicas de *dithering* mostradas na Seção 1 a todas as imagens que estão na pasta `i/`. Entretanto, é possível especificar a imagem e a técnica utilizadas executando-se `main.py -img IMAGE -t {0,1,2,3,4,5}`.

Pode-se visualizar todas as opções disponíveis com `main.py --help`.

3 Resultados

As imagens originais utilizadas nos resultados apresentados são mostradas na Figura 3.



Figura 3: Imagens utilizadas (todas com dimensão de 512×512 pixels)

Para todas as imagens foi utilizado um *threshold* de 128, ou seja, temos para cada pixel $[x, y]$ da imagem original ($0 \leq x < \text{largura}$, $0 \leq y < \text{altura}$):

$$\text{pixel}[x, y, z] = \begin{cases} 0 & \text{se } \text{pixel}[x, y, z] < 128 \\ 255 & \text{caso contrário} \end{cases}$$

onde em imagens coloridas $z \in [0, 3)$ representa as bandas de cor, e em imagens monocromáticas $z = 0$ (pois temos apenas uma banda).

Além da forma clássica de percorrer imagens *raster* (de cima-para-baixo, da esquerda-para-direita), implementou-se a varredura *serpentine* que alterna a orientação horizontal a cada linha para minimizar padrões indesejados como a impressão de certa direcionalidade [5].

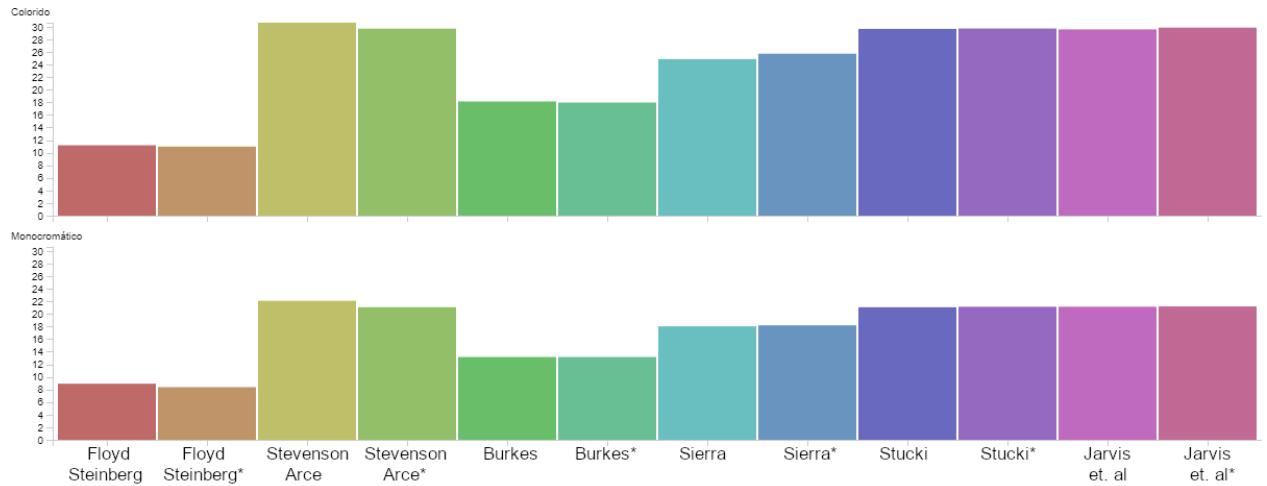


Figura 4: Tempos de execução (em segundos) para a imagem `peppers.png`. As técnicas com asterisco (*) utilizaram a varredura *serpentine*, ao invés do usual padrão *raster*

A Figura 5 apresentada os resultados da aplicação das diferentes transformações na imagem `baboon.png`, em cores e em níveis de cinza.

Já na Figura 6, destaca-se um detalhe da imagem `lenna.png` monocromática, onde podemos mais facilmente observar os artefatos gerados por cada técnica utilizada, e comparar o efeito das varreduras *raster* e *serpentine*.

Destaca-se que na varredura *serpentine*, ao passarmos da direita-para-esquerda as matrizes de difusão são espelhadas.

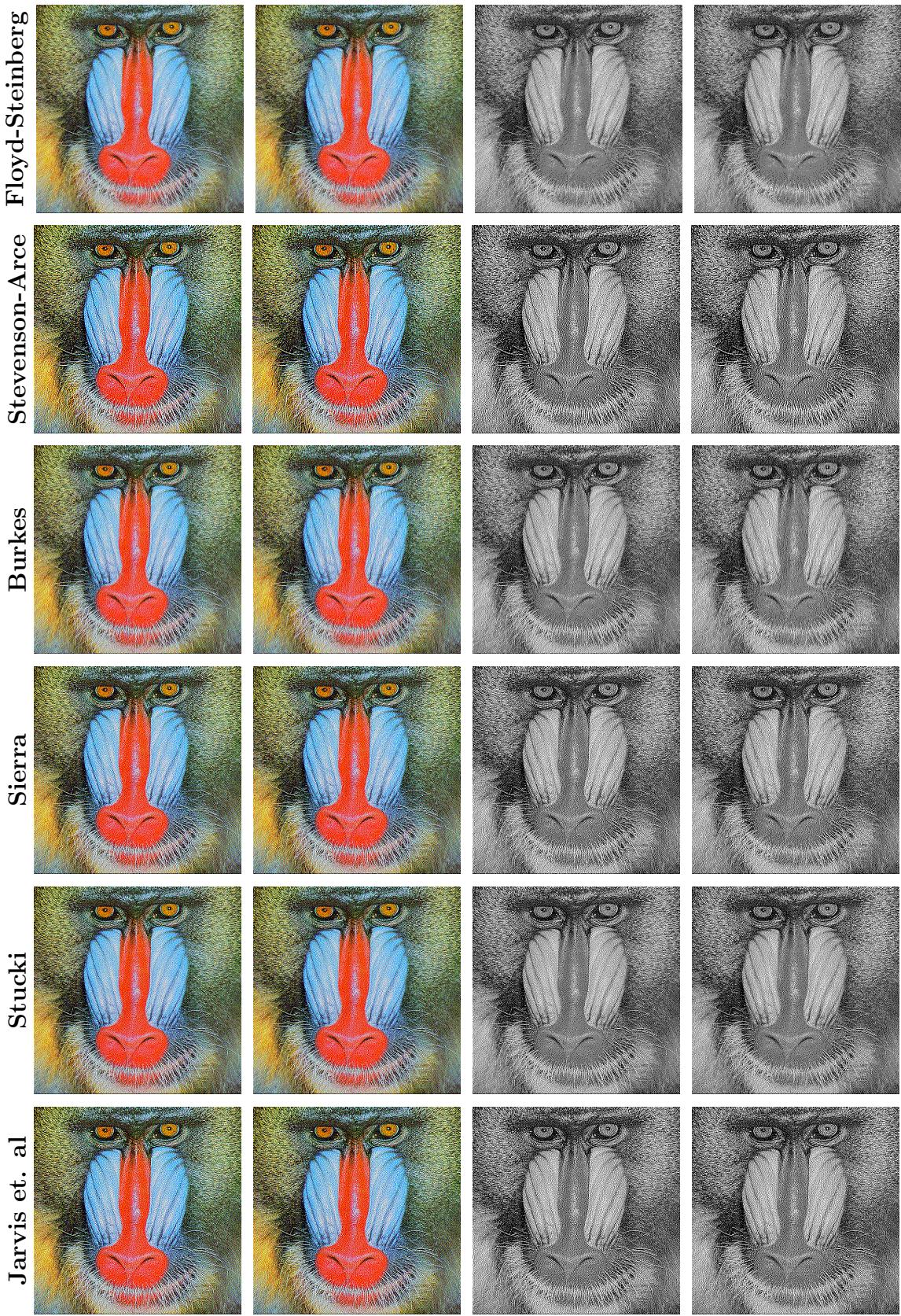


Figura 5: Imagens resultantes de técnicas de *dithering* com difusão de erro. A 1^a e 3^a coluna mostram a varredura *raster*, enquanto a 2^a e 4^a mostram a varredura *serpentine*

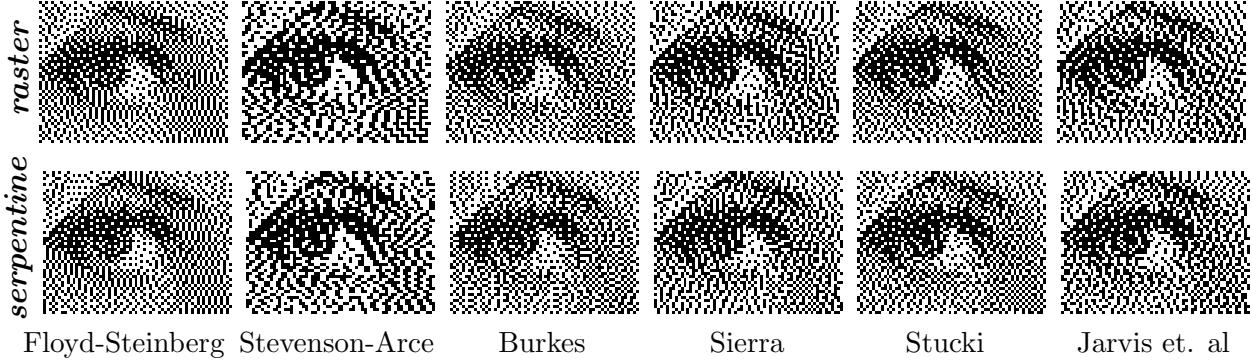


Figura 6: Detalhes de 64×48 pixels da imagem `lena.png` em níveis de cinza

4 Análise e Discussão

Dentre as seis técnicas apresentadas na Seção 1, a mais antiga e mais conhecida é a de Floyd e Steinberg, descrita em 1976 [6]. É importante notar que a matriz de difusão de erro de Floyd-Steinberg é também a menor delas (vide Figura 2), o que se reflete no tempo de execução das diferentes técnicas, computados na Figura 4, em que a vemos sendo a mais rápida.

No mesmo ano, Jarvis, Judice e Ninke publicaram seu kernel que distribui o erro por mais que o dobro de pixels, levando cerca de duas vezes mais tempo do que a técnica de Floyd-Steinberg para executar. Comparando seus resultados das Figuras 5 e 6, as imagens em meios-tonos de Jarvis et. al apresentam uma "granularidade" maior nos pixels. Assim, a uma distância maior do que das imagens de Floyd-Steinberg, percebemos que as de Jarvis não tem realmente um degradê de tons, ou seja, seu resultado é inferior.

A forma de Stucki foi publicada cinco anos depois [7] e tem o mesmo tamanho da de Jarvis et. al, consequentemente, ambas levaram o mesmo tempo de execução (21 segundos em uma imagem monocromática de 512×512), porém não observamos diferença na qualidade.

Sete anos após Stucki, Burkes publicou a distribuição mostrada na Figura 2e com o objetivo de melhorar a técnica de Jarvis, removendo uma linha de sua matriz, e então sendo a segunda técnica mais rápida. Entretanto, há uma impressão de certa direcionalidade em relação à aplicação de Jarvis (p.e. compare o nariz do `baboon.png` na Figura 5).

Em 1985 Stevenson e Arce publicaram sua abordagem [8], que distribui o erro na mesma quantidade de pixels que Stucki e Jarvis (12 pixels), porém em uma área maior, utilizando uma matriz 4×7 ao invés de uma 3×5 . Dentre todas as técnicas, essa foi a que resultou na maior "granularidade" dos pixels (veja a Figura 6), e assim, na pior qualidade.

Por último, a técnica de Sierra é a mais recente. Em 1989 Sierra divulgou três algoritmos [3], sendo o mostrado na Figura 2d também conhecido como Sierra-3. Por utilizar 10 pixels, sua execução é um pouco mais rápida que a das formas que alteram 12 pixels, mas não há uma piora significativa de qualidade, fazendo dele uma boa alternativa para os filtros de Stucki, Stevenson-Arce e Jarvis, Judice e Ninke.

Finalmente, ao compararmos a aplicação dos filtros apenas da esquerda-para-direita com a varredura *serpentine*, vemos que a segunda resulta em menos artefatos, principalmente em regiões de cor sólida, e, ao mesmo tempo, não leva a diferenças significativas de tempo de execução (conforme Figura 4). Portanto, devemos sempre optar por ela para técnicas de meios-tonos de imagens gerais.

5 Conclusões

Apesar de ser a forma mais antiga, a técnica de Floyd e Steinberg além de ter sido a mais rápida apresentou bons resultados, evidenciando por que é até hoje a mais conhecida.

Analisando subjetivamente os resultados das três imagens da Figura 3, as melhores técnicas foram Sierra e Jarvis et. al. Entretanto, como Sierra atualiza 10 pixels ao invés de 12, ele foi mais rápido. Nota-se ainda que seu divisor é múltiplo de 2 (enquanto o de Jarvis não é), de forma que uma implementação específica pode ser ainda mais rápida ao utilizar operações *bitwise* ao invés de divisões numéricas (*right shift*).

Portanto, se pudermos gastar um pouco mais de tempo, o método de meios-tonos com difusão de erro de Sierra é a melhor opção.

Por fim, uma melhora que pode ser aplicada a todas as técnicas é a utilização da varredura *serpentine* (alternância da direção horizontal a cada linha percorrida), que não aumentou o tempo de execução e diminuiu artefatos visuais, principalmente em regiões de cor sólida.

Referências

- [1] “Halftone — Wikipedia, The Free Encyclopedia,” 2019, acessado em setembro de 2019. [Online]. Available: <https://en.wikipedia.org/wiki/Halftone> 1
- [2] Hélio Pedrini, “Realce,” acessado em setembro de 2019. [Online]. Available: http://www.ic.unicamp.br/~helio/disciplinas/MC920/aula_realce.pdf 1
- [3] “Dither — Wikipedia, The Free Encyclopedia,” 2019, acessado em setembro de 2019. [Online]. Available: <https://en.wikipedia.org/wiki/Dither> 1, 5
- [4] “Ordered dithering — Wikipedia, The Free Encyclopedia,” 2019, acessado em setembro de 2019. [Online]. Available: https://en.wikipedia.org/wiki/Ordered_dithering 1
- [5] L. D. Crocker, P. Boulay, and M. Morra, “DHALF.TXT,” acessado em setembro de 2019. [Online]. Available: <https://github.com/SixLabors/ImageSharp/blob/master/src/ImageSharp/Processing/Processors/Dithering/DHALF.TXT> 3
- [6] R. W. Floyd and L. S. Steinberg, “An adaptive algorithm for spatial gray scale,” 1975. 5
- [7] T. Helland, “Image dithering: Eleven algorithms and source code,” acessado em setembro de 2019. [Online]. Available: <http://www.tannerhelland.com/4660/dithering-eleven-algorithms-source-code/> 5
- [8] R. L. Stevenson and G. R. Arce, “Binary display of hexagonally sampled continuous-tone images,” pp. 1009–1013, Jul 1985. 5