Relatório - Trabalho 02

MO443 - Introdução ao Processamento de Imagem Digital

Vinicius Teixeira de Melo - RA: 230223

Universidade Estadual de Campinas

viniciusteixeira@liv.ic.unicamp.br

8 de Maio de 2019

I. Especificação do Problema

O objetivo deste trabalho é implementar duas técnicas de quantização de cores, definidas como técnicas de pontilhado que visam reduzir a quantidade de cores utilizadas para exibir uma imagem, procurando manter uma boa percepção por parte do usuário.

O trabalho é dividido em duas partes. A primeira é escrever uma função para alterar os níveis de cinza $[f_{min} \dots f_{max}]$ de uma imagem $f(x, y)$ por meio das técnicas de pontilhado (*half-toning*) ordenado, produzindo uma imagem $g(x, y)$.

O algoritmo de pontilhado ordenado utiliza um conjunto de padrões formados por pontos pretos e brancos. O conjunto de dez padrões, ilustrados na Figura 1, pode ser representado por meio da matriz M_{3x3} a seguir:

$$M_{3x3} = \begin{array}{|c|c|c|}\hline 6 & 8 & 4 \\ \hline 1 & 0 & 3 \\ \hline 5 & 2 & 7 \\ \hline \end{array}$$

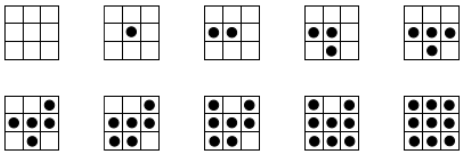

Figura 1: *Dez padrões de 3 x 3 pixels.*

Os valores das células da matriz podem ser utilizados como limiares. Se o valor do pixel (normalizado entre 0 e 9) for menor do que o número correspondente à célula da matriz, o pixel será substituído pelo valor preto, caso contrário, será substituído pelo valor branco.

Uma outra máscara, conhecida como matriz de pontilhado ordenado de Bayer, é dada por

$$M_{4x4} = \begin{array}{|c|c|c|c|}\hline 0 & 12 & 3 & 15 \\ \hline 8 & 4 & 11 & 7 \\ \hline 2 & 14 & 1 & 13 \\ \hline 10 & 6 & 9 & 5 \\ \hline \end{array}$$

A segunda parte do trabalho é aplicar 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 consideração 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 2):
   (a) adicione $\frac{7}{16}$ do erro ao pixel localizado à direita.
   (b) adicione $\frac{3}{16}$ do erro ao pixel localizado abaixo e à esquerda.
   (c) adicione $\frac{5}{16}$ do erro ao pixel localizado abaixo.

(d) adicione $\frac{1}{16}$ do erro ao pixel localizado abaixo e à direita.

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

Figura 2: *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 3(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 3(b)).

Figura 3: *Formas de varredura da imagem.*

Portanto, o trabalho é a aplicação dessas duas técnicas de pontilhado.

II. Entrada de Dados

O código fonte criado para a execução de todas as tarefas está no notebook **Trabalho 02.ipynb**. O código foi criado para aceitar imagens em tons de cinza do tipo PGM (*Portable GrayMap*).

Para executar o notebook, basta iniciar o ambiente *Jupyter Notebook*, abrir o notebook **Trabalho 02.ipynb** e executar as células em ordem. Todo o algoritmo foi implementado na linguagem Python na versão 3.6.

As imagens de entrada utilizadas nos testes do algoritmo foram retiradas da página do prof. Hélio Pedrini: Imagens. Na pasta **imgs/** estão as três imagens monocromáticas utilizadas nos testes: **baboon.pgm**, **fiducial.pgm** e **monarch.pgm**. As dimensões das imagens de entrada utilizadas são 512x512, 640x480 e 768x512, respectivamente.

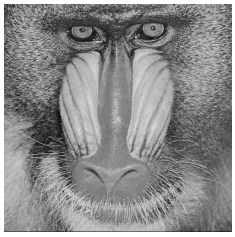

Figura 4: *baboon.pgm*

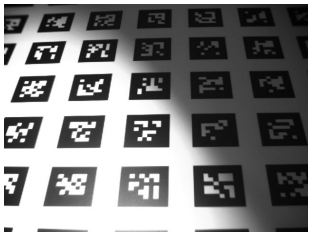

Figura 5: *fiducial.pgm*

Figura 6: *monarch.pgm*

III. Dependências e Códigos

As bibliotecas utilizadas neste trabalho foram:

Biblioteca	**Versão**
numpy	1.16.2
cv2	3.4.2
matplotlib	3.0.3
warnings	2.1

A leitura das imagens foi realizada utilizando uma função do **opencv** [1] chamada

imread(), a qual necessitou de uma constante do próprio **opencv** para que a imagem ficassem apenas com o canal de escala de cinza, essa constante é denominada **IMREAD_GRAYSCALE**.

As máscaras foram gerados manualmente através da função **numpy.array()**. A função para normalizar os níveis de cinza foi feita aplicando a fórmula de mudança de base descrita a seguir:

$$g(x) = \left(\frac{g_{max} - g_{min}}{f_{max} - f_{min}}\right)(f(x) - f_{min}) + g_{min}$$

A função de aplicação da técnica de pontilhamento *half-toning* foi feita manualmente, aplicando primeiramente a normalização de intensidade dos pixels, e depois o preenchimento da nova imagem com base nos valores que eram obtidos através da utilização da máscara como *threshold*.

As funções para a aplicação da técnica de pontilhamento com difusão de erro de *Floyd-Steinberg* também foram feitas manualmente, a qual a primeira, **floyd_steinberg_order1**, percorre a imagem como mostrado na Figura 3(a) e a segunda, **floyd_steinberg_order2**, percorre a imagem como mostrado na Figura 3(b).

IV. Fundamentação

i. Pontilhado Ordenado

Também conhecido como *Halftone* (meio-tom), o pontilhado ordenado é uma técnica de processamento de imagens que emprega padrões de pontos pretos ou brancos para reduzir o número de níveis de cinza de uma representação. Defivo ao fato do sistema visual humano atenuar a distinção entre os pontos com tons diferentes, os padrões de pontos pretos e brancos produzem um efeito visual como se a imagem fosse composto por mais tons de cinza [4].

No caso de uma máscara $3x3$, cada conjunto de níveis de cinza é representado por um padrão $3x3$ de pontos brancos ou pretos. Uma área de $3x3$ pixels cheia de pontos pretos é uma aproximação de um nível de cinza preto (ou 0). De forma análoga, uma área $3x3$ de pontos totalmente brancos será uma aproximação do branco em uma escala de nível de cinza (ou 9). Os demais padrões intermediários entre $]0, 255[$ serão re-escalados para o intervalo $[0, 9]$, conforme ilustrado na Figura 7.

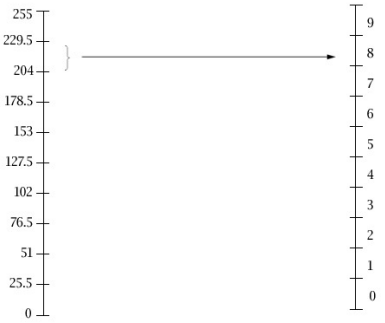

Figura 7: *Transformação de escala com uma máscara* $3x3$.

É possível notar que utilizamos nove limiares (*threshold*) para a sub-codificação: o intervalo $[0, 25.5[$ é truncado para 0, $[25.5, 51[$ é truncado para 1, $[51, 76.5[$ para 2, e assim por diante. Assim, cada pixel na imagem de entrada irá corresponder a um padrão de $3x3$ pixels na imagem gerada. Desta forma, a resolução espacial será reduzida a 33% da resolução espacial original. Isto é, antes da substituição de cada pixel pela máscara devemos reduzir em $\frac{1}{3}$ as dimensões da imagem para que o resultado tenha o mesmo tamanho da original [4].

ii. Pontilhado com Difusão de Erro

Pontilhado com difusão de erro é uma técnica de *halftone* que procura distribuir a diferença entre o valor exato de cada pixel e seu valor aproximado a um conjunto de pixels adjacentes. Diversas técnicas foram desenvolvidas com esta proposta, sendo o algoritmo de Floyd-Steinberg o primeiro desenvolvido utilizando esta abordagem [5].

O algoritmo varre a imagem da esquerda para a direita, de cima para baixo, quantificando os valores de pixel um por um. O erro

de quantização é transferido para os pixels vizinhos, embora não afete os pixels que já foram quantizados. Assim, se um número de pixels tiver sido arredondado para baixo, torna-se mais provável que o próximo pixel seja arredondado para cima, de modo que, em média, o erro de quantização seja próximo de zero.

Os coeficientes de difusão têm a propriedade de que, se os valores de pixel originais estiverem exatamente na metade entre as cores disponíveis mais próximas, o resultado pontilhado será um padrão de tabuleiro de damas. Por exemplo, 50% de dados em cinza podem ser pontilhados como um padrão quadriculado preto e branco. Para um pontilhamento ótimo, a contagem de erros de quantização deve ter precisão suficiente para evitar que os erros de arredondamento afetem o resultado [6].

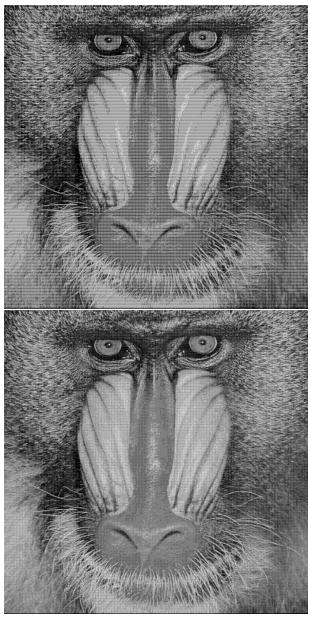

Figura 8: *Aplicação das máscaras M_{3x3} e M_{4x4}, respectivamente, na imagem* ***baboon.pgm****.*

V. Saída de Dados

Os imagens resultantes foram salvas dentro da pasta **resultados/** utilizando uma função da biblioteca **opencv** chamada **cv2.imwrite()** [1].

O formato dos nomes de saída estão da seguinte forma: para a técnica de pontilhado ordenado o padrão é o nome da imagem de entrada (*baboon*, *fiducial* e *monarch*) concatenado com o nome da máscara considerada para a aplicação da função sobre ela; para a técnica de pontilhado com difusão de erros são gerados duas imagens de saída para cada imagem de entrada, uma para cada forma de percorrer a imagem.

VI. Resultados e Discuções

Os resultados obtidos com a aplicação da técnica de pontilhado ordenado (*halg-toning*) estão a seguir:

A Figura 8 apresenta os resultados da aplicação na imagem **baboon.pgm**, o qual o resultado da esquerda sofreu um aumento nas dimensões da imagem de 300%, pois isso está diretamente ligado ao tamanho da máscara. Como a máscara é $3x3$, a imagem resultando possui 3 vezes mais *pixels* na dimensão x e y. A imagem resultante é formada também só por *pixels* pretos e brancos, pois cada *pixel* da imagem original resulta em "um *pixel*" do tamanho da máscara com valores 0 ou 255, porém no final, as imagens são transformadas em binárias para que cada *pixel* seja representado por um bit. A imagem que representa o resultado da aplicação da máscara M_{4x4} (imagem da direita), possui dimensoes 4 vezes maiores, por conta do tamanho da máscara. Como resutado visual, podemos observar que as imagens sofreram uma granularização dos *pixels* de acordo com o tamanho da máscara, é possível observar pequenos quadrados, proporcionais ao tamanho da máscara, na imagem. Então, podemos constatar que quanto maior for a máscara, maior será a imagem resultante e menor será a percepção desses quadrados,

se as imagens forem vistas com uma mesma dimensão final.

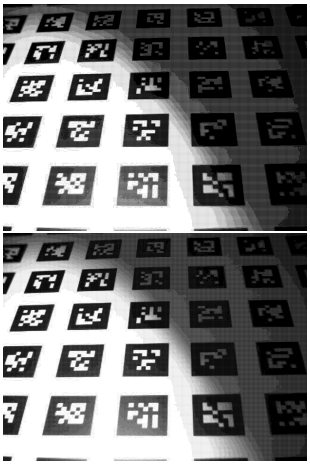

Figura 9: *Aplicação das máscaras M_{3x3} e M_{4x4}, respectivamente, na imagem **fiducial.pgm**.*

Na Figura 9, podemos constatar que o tamanho das imagens foi alterado também de acordo com o tamanho das máscaras, e como essa imagem de entrada possui menos detalhes que a anterior, pode-se observar melhor a diferença entre a aplicação das duas máscaras, a qual a máscara M_{4x4} torna a imagem resultante mais nítida que a aplicação com a máscara M_{3x3}.

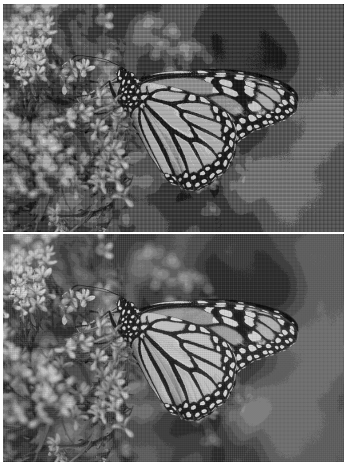

Figura 10: *Aplicação das máscaras M_{3x3} e M_{4x4}, respectivamente, na imagem **monarch.pgm**.*

Na Figura 10, podemos observar que algumas regiões sofreram um "agrupamento de intensidades" bem maior com a máscara M_{3x3} do que com a máscara M_{4x4}, a qual torna a imagem resultante com maior diferenciação visual das regiões, tendo assim um resultado melhor que o da aplicação da máscara M_{3x3}.

Os resultados obtidos com a aplicação da técnica de pontilhado com difusão de erro (*Floyd-Steinberg*) estão a seguir:

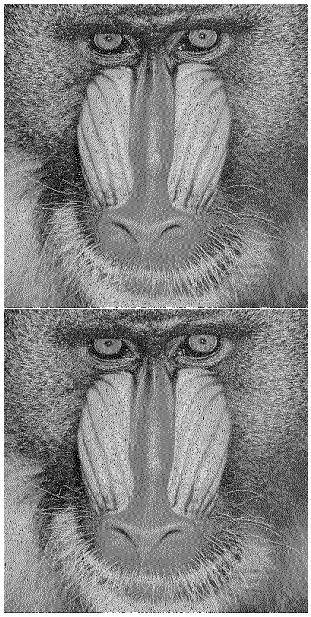

Figura 11: *Aplicação a técnica de pontilhado com difusão de erros percorrendo a imagem **baboon.pgm** das duas formas.*

A Figura 11 apresenta os resultados obtidos com a aplicação da técnica de pontilhado com difusão de erros, com duas formas de percorrer a imagem de entrada. As imagens resultantes não sofrem mudança nas suas dimensões, diferente da técnica anterior, e também são formadas apenas por *pixels* pretos e brancos. Como a imagem **baboon.pgm** possui um alto nível de detalhes, não é possível observar uma diferença significativa entre as duas formas de difundir o erro de cada *pixel*.

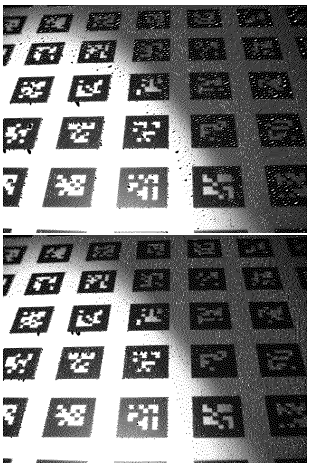

Figura 12: *Aplicação a técnica de pontilhado com difusão de erros percorrendo a imagem **fiducial.pgm** das duas formas.*

Na Figura 12, podemos observar que muitas regiões da primeira imagem possui algumas falhas, como conjuntos de *pixels* pretos em locais que era para ter *pixels* brancos, e vice-versa. Já na segunda imagem vemos que muitas das áreas defeituosas na primeira imagem estão consertadas aplicando a técnica de difusão de erro percorrendo de forma alternada, pois os erros são propagados nas direções esquerda-direita e direita-esquerda.

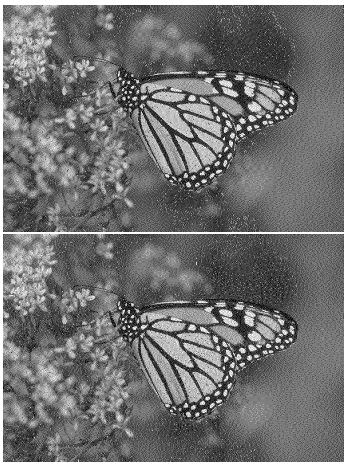

Figura 13: *Aplicação a técnica de pontilhado com difusão de erros percorrendo a imagem **monarch.pgm** das duas formas.*

Na Figura 13, podemos observar que em conjuntos de *pixels* que são limitantes de regiões com difentes intensidades, a aplicação da técnica percorrendo somente no sentido esquerda-direita, esses *pixels* tendem a ficar mais visíveis, porém, percorrendo de maneira alternada, as regiões de fronteiras tendem a ficar menos abruptas.

Com esses experimentos, podemos destacar a aplicação da técnica de difusão de erros de maneiras alternada obteve melhor resultado do que difudindo o erro somente em um sentido.

VII. Conclusão

Podemos concluir que os resultados obtidos com as aplicações das técnicas de pontilhado ordenado e pontilhado com difusão de erro foram satisfatórios quanto a especificação do trabalho e que a visualização prática da normalização da imagem, transformação para intensidade binária e transformações com pontilhados nas imagens consolida ainda mais os conceitos vistos em sala de aula.

Referências

[1] Welcome to opencv documentation! https://docs.opencv.org/2.4/index.html Acesso em: 04/05/2019.

[2] Pedrini, Hélio, and William Robson Schwartz. Análise de imagens digitais: princípios, algoritmos e aplicações. Thomson Learning, 2008.

[3] Matplotlib Version 3.0.3 https://matplotlib.org/contents.html Acesso em: 04/05/2019.

[4] SAWP, Dithering http://www.sawp.com.br/blog/?p=940 Acesso em: 04/05/2019.

[5] Digital Image Processing Using Python https://ckpytsan.wordpress.com/2011/01/14/dithering-parte-iii-dithering-com-difusao-de-erro-e-o-algoritmo-de-floyd-steinberg/ Acesso em: 04/05/2019.

[6] Wikipedia: Floyd–Steinberg dithering https://en.wikipedia.org/wiki/Floyd-Steinber$_d$*ithering* Acesso em: 04/05/2019.