Universidade Estadual de Campinas

Instituto de Computação

Introdução ao Processamento Digital de Imagem (MC920 / MO443)

Trabalho 2

Leodécio Braz - 230219

8 de maio de 2019

1 Especificação do Problema

O objetivo deste trabalho é, por meio das técnicas de **pontilhado ordenado** (*half-toning*) e **pontilhado com difusão de erro**, alterar os níveis de cinza de uma imagem de entrada $\mathbf{f}(x, y)$ gerando uma imagem de saída $\mathbf{g}(x, y)$

O algoritmo de pontilhado ordenado utiliza um conjunto de padrões formados por pontos pretos e brancos. Os valores das células da matriz podem ser utilizados como limiares. Se o valor do pixel normalizado, da imagem for menor do que o número correspondente à célula da matriz, o pixel será substituído pelo preto, caso contrário, será substituído pelo valor branco.

O algoritmo de pontilhado por difusão de erro também transforma uma imagem original em uma imagem contendo apenas os valores preto e branco, entretanto, leva em consideração os valores ao redor de cada pixel.

O problema consiste em as técnicas de pontilhado ordenado e a técnica de difusão de erro de Floyd-Steinberg em um conjunto de imagens. Para cada experimento realizado, mostrar a imagem original e as imagens resultantes pela aplicação de cada transformação.

2 Script

O script foi desenvolvido em Python na versão 3.0 e foi utilizado o framework Jupyter para documentação do mesmo. As bibliotecas utilizadas foram **OpenCV** [3], **Numpy**[2] e **matplotlib**[1].

2.1 Entrada

As imagens de entrada foram disponibilizadas pelo professor no endereço que se encontra em sua página[1]

[1] *http : //www.ic.unicamp.br/ helio/imagens_pgm/*

As imagens estão localizadas no diretório **./images/**, e já se encontram, em parte, declaradas no código.

2.2 Saída

Após a execução, o script irá gerar tanto resultados intermediários visíveis na tela, quanto as imagens resultante dos processos que serão salvas no diretório **./output/**.

As imagens de saída estão no formato PBM (*Portable BitMap*) e seus nomes seguem o padrão como na Tabela 1.

Tabela 1: Nome dos arquivos de saída

Nome do arquivo da imagem	**Descrição**
[imagem_entrada]_PO_mask[X]	Imagem de entrada após a aplicação do algoritmo de pontilhado ordenado com uma máscara (mask1 ou mask2)
[image_entrada]_PDE_ED	Imagem de entrada após a aplicação do algoritmo de pontilhado com difusão de erro com varredura da esquerda para a direita
[image_entrada]_PDE_BD	Imagem de entrada após a aplicação do algoritmo de pontilhado com difusão de erro e modificando a direção da varredura a cada linha

2.3 Leitura das imagens e declaração das máscaras

As imagens de entrada foram lidas através da função **imread** da biblioteca do OpenCV, que gera um *Array* do Numpy de dimensões MxN que corresponde ao tamanho da imagem.

As máscaras, também no formato de *Array* do Numpy, foram declarados manualmente no código. E são representadas como **mask1** e **mask2** a seguir:

$$mask1 = \begin{array}{|c|c|c|}\hline 6 & 8 & 4 \\ \hline 1 & 0 & 3 \\ \hline 5 & 2 & 7 \\ \hline \end{array} \qquad mask2 = \begin{array}{|c|c|c|c|}\hline 0 & 12 & 3 & 15 \\ \hline 8 & 4 & 11 & 7 \\ \hline 2 & 14 & 1 & 3 \\ \hline 10 & 6 & 9 & 5 \\ \hline \end{array}$$

3 Resolução

3.1 Técnica de pontilhado ordenado

Para a técnica de pontilhado ordenado, dada uma máscara e uma imagem de entrada de dimensões dxd e MxN respectivamente, primeiramente foi feito um

"rescale" na intensidade dos pixels da imagem, para que estes ficassem no intervalo definido de acordo com o tamanho da máscara que foi passada.

Após isto, criamos um nova imagem, inicialmente com 0's, e com dimensões $d*Mxd*N$ onde esta recebera os valores após o processo com a imagem original.

Percorremos cada pixel da imagem original e para cada pixel, chamamos uma função **update_mask** onde passamos o valor do pixel e a máscara de entrada. Essa função realiza a comparação se o valor do pixel for menor que um elemento da máscara, então substitui esse elemento por 0, caso contrário substitui por 1. Em seguida salvamos esse filtro com valores 0's e 1's na nova imagem que criamos.

Os resultados da aplicação desta técnica, com as máscaras **mask1** e **mask2** citadas anteriormente e diferentes imagens, estão descritos na Seção 4.

3.2 Técnica de difusão de erro de Floyd-Steinberg

Na técnica de difusão de erro, percorremos cada pixel da imagem original e para cada pixel, verificamos se seu valor é maior que 128, se sim então atualizamos seu valor para 255, se não atualizamos para 0.

Para cada pixel que atualizamos, obtemos também um valor de *erro* que corresponde à diferença do valor do pixel originalmente e o valor aproximado que o substituiu. Parcelas desse erro são então repassadas e adicionadas aos pixels vizinhos, como mostra a Tabela 2.

Tabela 2: Distribuição do erro

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

Na técnica de difusão de erro, percorremos a imagem de duas diferentes formas, a varredura da **esquerda para a direita** e outra, **modificando a direção de varredura a cada linha**. É importante ressaltar que ao modificar a direção da varredura, a distribuição de erro como mostrada na Tabela 2 é então espelhada.

Os resultados da aplicação destas técnicas estão descritos na Seção 4.

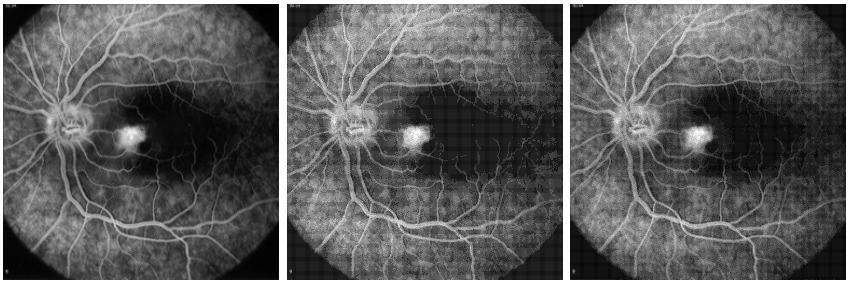
4 Resultados

Após a aplicação da técnica de pontilhado ordenado em um conjunto de imagens e com as duas máscaras declaradas, podemos ver os resultados na Figura 1.

Vale ressaltar que a técnica de pontilhado ordenado aumenta as dimensões da imagem, expandindo sua resolução. Nesse processo, algumas características podem passar a aparecer na imagem como no caso das Figuras 1b 1e, onde é possível notar uma espécie de "malha" contida na imagem gerada.

Figura 1: Aplicação da técnica de pontilhado ordenado

(a) Imagem Original (b) mask1 (c) mask2



(d) Imagem Original (e) mask1 (f) mask2

Da mesma forma que na técnica anterior, aplicamos a técnica de difusão de erro para um conjunto de imagens e com as diferentes varreduras. Os resultado podem ser vistos na Figura 2.

Nesse processo de pontilhado com difusão de erro, alguns padrões indesejados ou valores espúrios podem aparecer na imagem, como é possível notar aspectos parecidos com um ruido "sal e pimenta" nas Figuras 2b e 2c e aspectos parecidos com uma certa textura nas Figuras 2e e 2f.

Um comparativo entre as duas técnicas de pontilhado para uma mesma imagem pode ser visto da Figura 3. Em ambas as técnicas as imagens resultantes se assemelham à original e suas intensidades respeitam a propriedade de conter apenas valores 0's ou 1's.

Referências

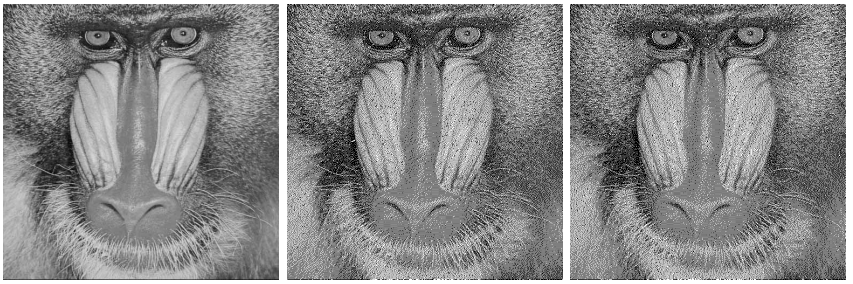
[1] Matplotlib user guide. https://matplotlib.org/contents.html. Accesso em: 05/05/2019.

[2] Numpy user guide. https://docs.scipy.org/doc/numpy/user/. Accesso em: 05/05/2019.

[3] Welcome to opencv documentation! https://docs.opencv.org/2.4/index.html. Accesso em: 05/05/2019.

Figura 2: Aplicação da técnica de pontilhado com difusão de erro

(a) Imagem Original (b) Esquerda-Direita (c) Modificando direção



(d) Imagem Original (e) Esquerda-Direita (f) Modificando direção

Figura 3: Resultados após a aplicação das técnicas de pontilhado

(a) Imagem Original (b) Pontilhado ordenado (c) Difusão de erro

(d) Imagem Original (e) Pontilhado ordenado (f) Difusão de erro