MC920 – Trabalho 1

Rafael Sartori M. Santos, 186154

11 de setembro de 2019

1 Introdução

O objetivo do trabalho é analisar, no processo de *half-toning*, os diferentes métodos de difusão de erro: Floyd-Steinberg, Sevenson-Arce, Burkes, Sierra, Stucki e Jarvis-Judice-Ninke. Aplicarei uma transformação em 2 níveis para cada camada de cor de uma imagem representada com 3 camadas: vermelho, verde e azul (R, G, B). Obteremos ao final 8 possibilidades de cor para cada ponto da imagem. Analisarei uma imagem de grande dimensão no formato PNG quanto a sua semelhança à original em uma tela de computador em 2 casos: curta e longa distância de visualização.

Farei esse processamento utilizando Python com as bibliotecas *OpenCV* e *NumPy* em um *Jupyter Notebook*.

2 Método

Capturei uma imagem de grande dimensão (3096x4128) colorida com meu celular sob condições de alta luminosidade, pois gostaria de avaliar o efeito produzido em imagens que são comuns ao dia a dia. Converti do formato original JPEG para PNG utilizando a ferramenta *GIMP*. No tratamento, utilizo apenas a PNG para não possuir intererência da compressão com perdas do outro formato.

Para realizar o processamento digital, as bibliotecas de Python que utilizei foram:

- *OpenCV* para abrir e salvar imagens;
- *NumPy* para aplicar transformações à imagem;
- Alguns módulos da padrão para calcular tempo de execução e iterar de formas diferentes na imagem.

Organizei todo código responsável pelo processamento e medição de tempo em um *Jupyter Notebook*; o que era responsável por abrir e salvar imagens e aplicar as conversões necessárias para utilizar nas bibliotecas em um módulo de utilidade (comum a outros trabalhos da disciplina).

2.1 Meios-tons

Escolhi manter apenas 2 níveis para cada camada da imagem. Como há 3 camadas de cores na imagem que testei, obtive 2^3 possibilidades de cores para cada ponto.

A aplicação de meios-tons na imagem f para produzir a imagem g é dada pela eq. (1).

$$g(x,y) = \begin{cases} 255 & \text{se } f(x,y) \geq 128 \\ 0 & \text{caso contrário} \end{cases} \tag{1}$$

2.2 Aplicação da difusão de erro

Represento cada método de difusão através de uma matriz, que chamarei simplesmente de filtro, cujo centro de aplicação deve ser mencionado para sabermos onde aplicá-lo de forma vetorial. A difusão de Floyd-Steinberg, por exemplo, é representada pelo par da matriz e centro de aplicação:

$$\left(\begin{bmatrix} 0 & 0 & 7/16 \\ 3/16 & 5/16 & 1/16 \end{bmatrix}, (0,1) \right)$$

2.3 Limitações

Como, para disseminar erros, é necessário alterar a imagem de entrada na aplicação, não é possível vetorizar toda a aplicação do filtro como é possível realizar em alguns casos. Por conta isso, só consegui vetorizar a aplicação da difusão de erro em cada ponto da imagem e a correção de valores (para manter a imagem no intervalo $[0, 255]$). Isso resulta em um código que demora vários minutos para imagens grandes: para a imagem que utilizei, o tempo de execução foi entre $879,98$ e $918,84$ segundos (aproximadamente 15 minutos).

Também não é possível executar testes automaticamente em imagens monocromáticas: é necessário possuir 3 camadas de cores na imagem. Apesar de ser fácil isolar no código o trecho de aplicação em uma única camada, não o fiz senão para testes enquanto desenvolvia.