Física Alternativa

Gubi

18 de outubro de 2016

Famos fazer um programa para distorcer imagens usando uma interação entre os pixels que obedece a uma física alternativa, muito diferente daquela que experimentamos no nosso particular universo.

Cada pixel da imagem possui as três componentes usuais: vermelho (R), verde (G) e azul (B). Cada uma possui um comportamento diferente, comandadas pela componente G, conforme descrito na seção 1.

1 Interação entre os pixels

As componentes R e B são repelidas pela componente G em direções opostas. Vamos considerar cada cor como um valor no intervalo $[0, 1[$, correspondendo à intensidade desta componenten no pixel, como usual.

Adicionalmente, o valor de G indica um ângulo, contado a partir do eixo vertical e no sentido horário, com um mapeamento direto para intervalo $[0, 2\pi[$, isto é $\theta_G = 2\pi G$.

R e B correspondem alinhados com a direção indicada por G, sendo que B aponta na direção oposta. As normas dos vetores correspondem à intensidade de cada componente de cor.

A figura 1 mostra a disposição dos vetores e da componente G, para o pixel $(.7, .17, .4)$.

1.1 Interações

Para simplificar o cálculo, cada pixel interage apenas com seus primeiros vizinhos (acima, abaixo, direita e esquerda). A intensidade da interação depende das projeções de R e B nos eixos.

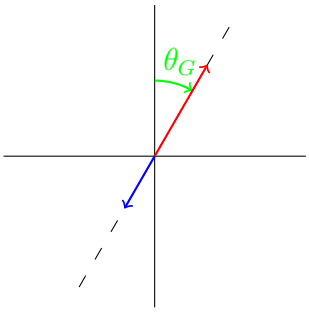

Figura 1: Componentes de cor

Para a componente de cor C^{ij} do pixel (i, j), onde C pode ser R ou B (a cor verde será tratada a seguir), sejam (C_x^{ij}, C_y^{ij}) suas componentes horizontal e vertical, respectivamente.

Uma parte do valor de C^{ij} será transferida para os vizinhos que se encontram na mesma direção. Por exemplo, se $C_x^{ij} > 0$, parte do seu valor será acrescido a $C_x^{i+1,j}$, caso contrário, a trasnferência se dará a $C_x^{i-1,j}$. O procedimento é análogo para a outra componente.

O valor transferido depende do vizinho que o recebe, segundo a seguinte fórmula:

$$\Delta = \frac{(1 - C^{Vizinho}) \times C_d^{ij}}{4} \tag{1}$$

Após *todas* as transferências, cada pixel deve ser verificado e eventualmente corrigido para que não possua nenhuma cor com valor maior que 1 menor do que 0. Quando um valor passa de 1., o excedente deve ser redistribuído entre os vizinhos uniformemente, desde que isto não provoque novo estouro.

A componente Gé atualizada de acordo com os valores obtidos após a atualização de R e B. Considere o vetor (R, B), o ângulo que este vetor forma com a vertical, no sentido horário, deve ser somado a θ_G e G deve ser corrigido de acordo.

Nota: As bordas são consideradas fixas e nunca se alteram.

2 O programa

O programa deve receber o seguinte conjunto de parâmetros de na linha de comando, nesta ordem:

1. Nome do arquivo de entrada
2. Nome do arquivo de saída
3. Número de iterações
4. Número de processadores

3 Saída

A saída será formada pelo arquivo com a imagem modificada, no formato *PPM*, tipo `P3` (colorido ASCII). A descrição do formato se encontra a seguir.

3.1 Formato *PPM*

O formato é bastante simples:

- A primeira linha contém o tipo da imagem, com dois caracteres:

  P1 Bitmap, em modo ascii
  P2 *Greyscale*, modo ascii
  P3 Colorida, modo ascii
  P4 Bitmap, binária
  P5 *Greyscale*, binária
  P6 Colorida, binária

  As linhas seguintes podem começar com '`#`' e servem como comentários.

  Em seguida vem um par de inteiros representando a largura e a altura da imagem. O próximo inteiro indica o valor máximo de uma componente, no nosso caso, será 255.

  Em seguida são colocados os valores de cada pixel. Para o formato *P3* são triplas de inteiros, com as componentes RGB. No *P6* são triplas de *bytes*. Para mais informações, procure pelo formato `Netpbm` no *Google* ou similar.

Que a velocidade esteja com vocês.