Operações Aritméticas e Transformações Geométricas

Guilherme Brandt[1], Heitor Galdino[1], Hernandes Macedo[1], Thaís Calixto[1]

[1]Ciência da Computação – Universidade Federal do Tocantins (UFT)
Palmas – TO – Brasil

guilhermebrandt@mail.uft.br, heitor.galdino@mail.uft.edu.br,
hernandes.macedo@mail.uft.edu.br, thais.calixto@mail.uft.edu.br

Abstract. *This paper touches on the usage of arithmetic operation and geometrical transformation algorithms for digital image manipulation.*

Resumo. *Este artigo discorre sobre a utilização de algoritmos que realizam operações aritméticas e transformações geométricas para a manipulação de imagens digitais.*

1. Introdução

Imagens digitais baseadas em Raster (lit. *Varredura*) são mapas de bits bidimensionais que armazenam descrições em cada uma das coordenadas de seu mapa. Em imagens sem compressão, descrevem matrizes bidimensionais onde o menor ponto de informação relevante é um pixel, uma unidade de informação que contém valores de cor e, em alguns formatos, transparência. Por serem matrizes de informação, estão sujeitas a operações matemáticas que realizem transformações matriciais.

O propósito deste artigo é explorar as operações aritméticas e transformações geométricas que podem ser aplicadas a imagens baseadas em Raster, como adição, subtração, multiplicação e divisão entre imagens, bem como rotações, translações, escala e espelhamento. O escopo deste artigo também contempla operações pontuais, locais e operações lineares e não lineares.

2. Referencial Teórico

2.1 Tipos de operações

2.1.1 Operações aritméticas e geométricas

Operações aritméticas são operações realizadas com base em pixels individuais entre imagens de bandas diferentes através de uma regra matemática definida, tendo como resultado uma banda representando a combinação das bandas originais. Permitem comprimir dados, mas podem acarretar na perda de informações originais quando os resultados extrapolarem o intervalo de 0 a 255.

Transformações geométricas são todas as transformações responsáveis por mudança na orientação, tamanho e formas dos objetos, alterando os valores das coordenadas que os descrevem.

2.2 Região de operações

2.2.1 Operações pontuais e locais

Operações pontuais são um método de processamento de imagens no qual cada pixel da saída é unicamente dependente do pixel correspondente na imagem de entrada, independente de sua localização ou de seus pixels vizinhos.

Operações locais são operações onde um pixel individual sofre influência de seus vizinhos. São capazes de procurar formas na imagem através de padrões de busca, definir bordas na imagem, remover ruído, etc.

2.3 Linearidade da operação

2.3.1 Operações lineares e não lineares

Operações lineares de processamento de imagens são baseadas nas mesmas técnicas convencionais de processamento de sinais digitais: Convolução e Análise de Fourier. Filtragem linear pode ser utilizada para ressaltar bordas de objetos, reduzir ruídos, corrigir desequilíbrios na iluminação, etc. São definidos por uma variedade de princípios, como, por exemplo, a definição básica de linearidade. Se um sistema é definido como tendo uma entrada descrita por x[n] = ax[n1] + bx[n2], a resposta de um sistema linear é da natureza de y[n] = ay[n1] + by[n2]. Isso é conhecido como a propriedade de superposição.

Operações não-lineares não seguem a natureza de saída de operações lineares e são capazes de produzir resultados de uma maneira pouco intuitiva. Um exemplo de filtro não-linear é o filtro de mediana, onde sua saída depende da ordenação dos valores de entrada, normalmente ordenados em ordem crescente ou decrescente.

3. Operações Aritméticas

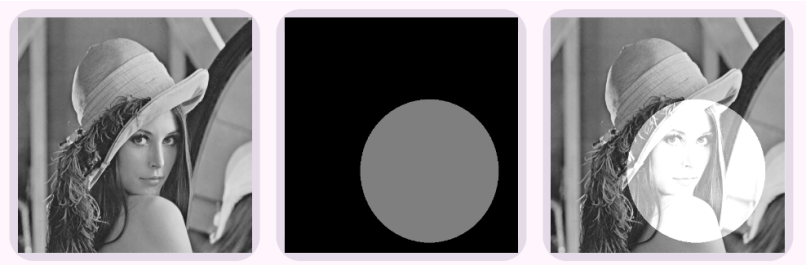
3.1 Soma e subtração

As operações de soma e subtração são análogas e responsáveis por somar ou subtrair duas imagens ao operar o valor da luminância de cada pixel e prender o resultado ao intervalo {0; 255}.

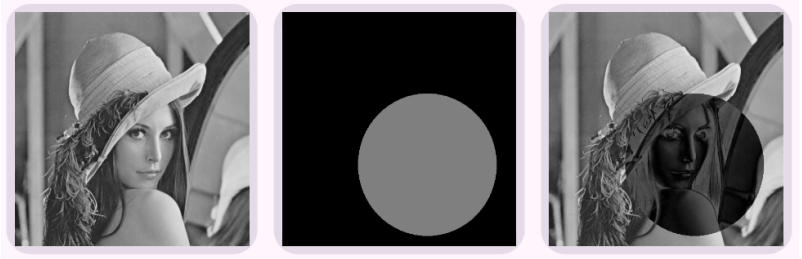
Trecho do código da operação de soma:

```
for (var i = 0; i < pixelsImgA.length; i++) {
  final sum = pixelsImgA[i] + pixelsImgB[i];
  resultImagePixels[i] = sum.clamp(0, 255);
```

Resultado da operação de soma:



Resultado da operação de subtração:



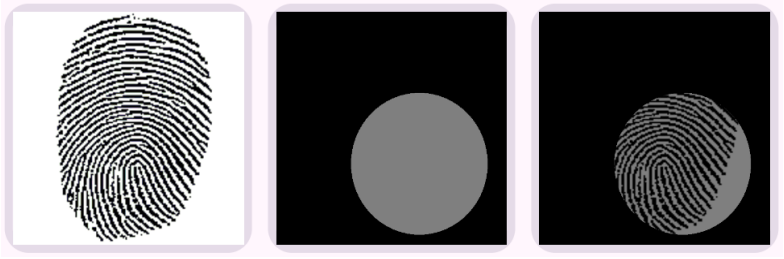
3.2 Multiplicação

Responsável por multiplicar a luminância entre duas imagens, pixel a pixel. Seus valores de luminância são divididos por 255 antes da multiplicação, com 0 representando luminância 0 e 1 representando luminância 255.

Trecho do código da operação de multiplicação:

```
for (var i = 0; i < pixelsImgA.length; i++) {
    final mult = (pixelsImgA[i] / 255) * (pixelsImgB[i] / 255);
    resultImagePixels[i] = (mult * 255).toInt();
```

Resultado da operação de multiplicação:



3.3 Divisão

Responsável pela divisão entre os valores de luminância dos pixels da imagem A e B, um a um. Se o pixel de B tiver luminância 0, toma 1 como denominador. Como os resultados são muito próximos uns dos outros, uma normalização min-max é realizada para melhor distribuir os valores no intervalo de luminância {0; 255}.

Trecho do código da operação de divisão:

```
for (var i = 0; i < pixelsImgA.length; i++) {
    results.add(pixelsImgB[i] != 0
        ? (pixelsImgA[i] / pixelsImgB[i])
        : (pixelsImgA[i] / 1));
```

Resultado da operação de divisão:

4. Transformações Geométricas

4.1 Translação

Responsável por mover a matriz de pixels da imagem em um ou dois eixos em uma certa quantia. É realizada pela multiplicação de seus valores por um vetor com os valores a serem transladados.

Trecho do código da operação de translação:

```
return List.generate(
    data['height']!,
    (y) => List.generate( data['width']!,
      (x) { try {
          return imageMatrix[y - data['moveY']!][x -
data['moveX']!];
```

Resultado da operação de translação:

4.2 Rotação

Responsável por multiplicar a matriz de pixels da imagem por uma matriz de rotação correspondente ao ângulo de rotação desejado. A matriz de rotação é dada por:

$$R = \begin{matrix} cos\theta & -sen\theta \\ sen\theta & cos\theta \end{matrix}$$

A equação utilizada envolvendo newX e newY no código vem da multiplicação das matrizes ao desenvolver o produto de X, Y genéricos pela matriz de multiplicação genérica. Tem seu ponto de pivô no canto superior esquerdo.

Trecho do código da operação de rotação:

```
for (var y = 0; y < data['height']!; y++) {
    for (var x = 0; x < data['width']!; x++) {
      final newX = x * cos(rads) + y * sin(rads);
      if (newX < 0 || newX >= data['width']!) continue;
      final newY = y * cos(rads) - x * sin(rads);
      if (newY < 0 || newY >= data['height']!) continue;
      newImage[newY.toInt()][newX.toInt()] = imageMatrix[y][x];
```

Resultado da operação de rotação:

4.3 Escala

Responsável por escalar os pixels de uma imagem dado um fator numérico. É alcançada através da multiplicação por uma matriz cujos elementos da diagonal principal determinam o fator de escala em cada eixo:

$$S = \begin{matrix} Cx & 0 & 0 \\ 0 & Cy & 0 \\ 0 & 0 & 1 \end{matrix}$$

Trecho do código da operação de escala:

```
(x) {
        try {
          return imageMatrix[y ~/ (data['scale']! / 10)]
              [x ~/ (data['scale']! / 10)];
```

Resultado da operação de escala:

4.4 Reflexão

Responsável por espelhar os pixels de uma imagem em um dado eixo. A matriz que realiza a transformação em questão se dá por:

$$\begin{matrix} x' \\ y' \\ 1 \end{matrix} = \begin{matrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{matrix} \begin{matrix} x \\ .y \\ 1 \end{matrix}$$

Trecho do código da operação de reflexão:

```
final yCoord =
            data['reflectionType'] != 1 ? (y - (data['height']!
- 1)).abs() : y;
        final xCoord =
            data['reflectionType'] != 2 ? (x - (data['width']! -
1)).abs() : x;
```

Resultado da operação de reflexão:

5. Conclusão

A natureza matricial de rasters permite que diversas operações matemáticas sejam utilizadas para transformações na imagem por completo, como as modificações básicas de rotação, translação, escala e reflexão, bem como operações baseadas nestas (como escala em direções variadas causando um efeito de skewing, por exemplo).

Algoritmos que necessitam trabalhar uma imagem em base de pixels individuais devem ser tratados com cuidado no quesito de desempenho, visto que uma operação simples de multiplicação possa envolver diversos laços de repetição aninhados causam um impacto considerável ao depender da plataforma onde estão sendo executados.

Referências

Gonzalez, R. e Woods, R. (2009) “Processamento Digital de Imagens”. Pearson São Paulo, 3ª edição.

Spring (2019) “Teoria: Processamento de imagens”. Divisão de processamento de imagens, INPE.

Arakaki, J. (2018) “Computação Gráfica e Processamento de Imagens: Transformações geométricas (2D)”. PUC São Paulo.

Cámara-Chávez, G. (2017) “Operações Pontuais”. Departamento de Computação, Universidade Federal de Ouro Preto.

College Friendly (2021) “Point operations in digital image processing with examples”. Youtube.

Saberi, A. (2013) “Digital image processing: p018 Introduction to local neighborhood operations”. Youtube.

Smith, S. W. (1997) “The Scientist and Engineer's Guide to Digital Signal Processing”.

Rush, A. (2012) “Comparing linear versus nonlinear filters in image processing”. Embedded Computing Design.