Processamentos e Transformações em Imagens

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, as well as image histograms and image filters.*

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

1 Operações aritméticas e transformações geométricas

1.1 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 capítulo é explorar operações aritméticas e transformações geométricas que podem ser aplicadas a imagens baseadas em Raster, como adição, subtração, multiplicação e divisão, bem como rotações, translações, escala e espelhamento. Também contempla operações pontuais, locais e operações lineares e não- lineares.

1.2 Referencial Teórico

1.2.1 Tipos de operações

1.2.1.1 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.

1.2.2 Região de operações

1.2.2.1 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.

1.2.3 Linearidade da operação

1.2.3.1 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.

1.3. Operações Aritméticas

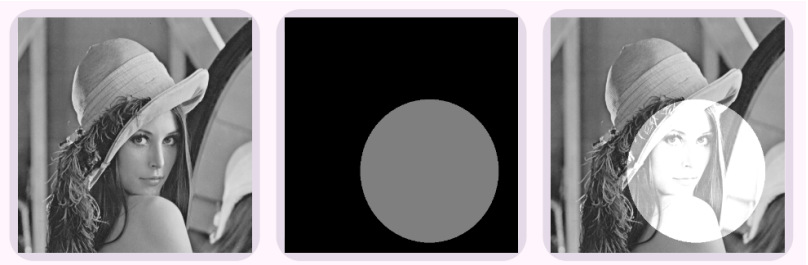
1.3.1 Soma e subtração

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}.

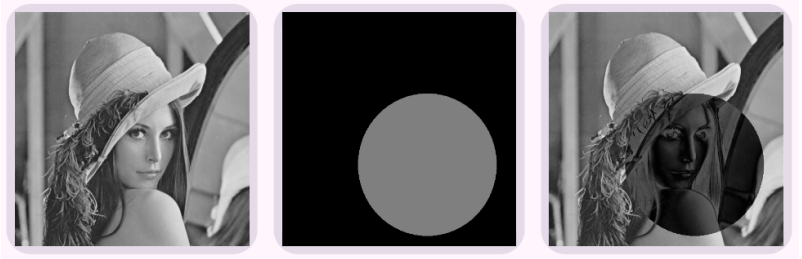
Trecho do código da operação de soma:

```
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:



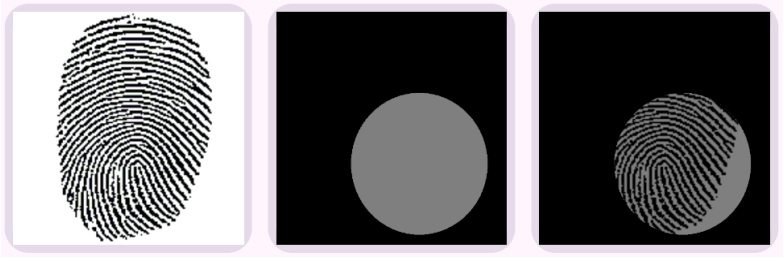
1.3.2 Multiplicação

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:



1.3.3 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:

1.4. Transformações Geométricas

1.4.1 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:

1.4.2 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:

1.4.3 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:

1.4.4 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:

1.5 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.

2 Processamento de Histogramas

2.1 Introdução

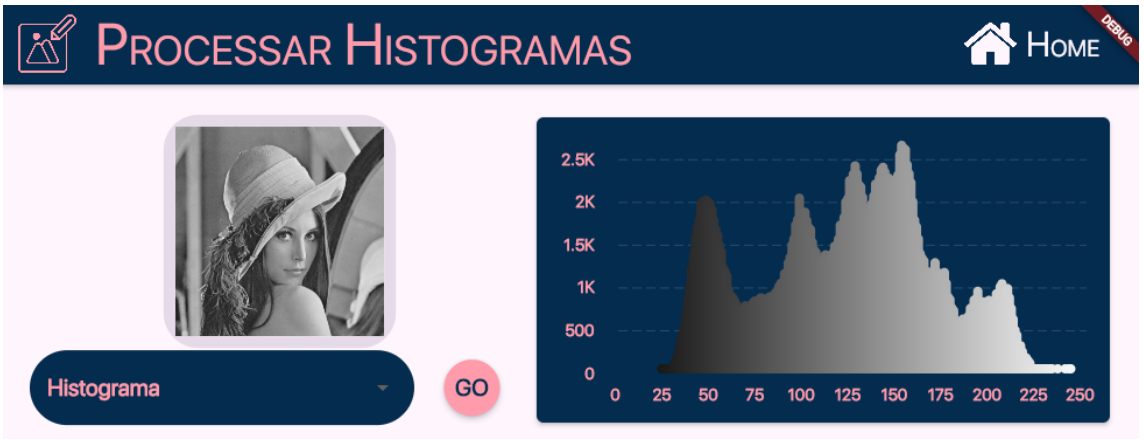
Histogramas de imagens representam a frequência com a qual um valor de intensidade ocorre numa imagem. Normalmente é representado por um gráfico de barras bidimensional, com o eixo X representando os elementos de intensidade e o eixo Y representando a frequência de cada intensidade. Comumente são utilizados para a realização de ajustes na exposição e/ou contraste de uma imagem.

O propósito deste capítulo é estudar a forma que histogramas são obtidos e conhecer os ajustes possíveis a partir da manipulação das informações obtidas através de um histograma.

2.2 Tipos de histogramas

2.2.1 Histograma básico

Um histograma básico é um histograma de uma imagem com seus dados originais, sem nenhum ajuste em sua curva de intensidade cumulativa. Contém os valores literais da imagem de entrada.

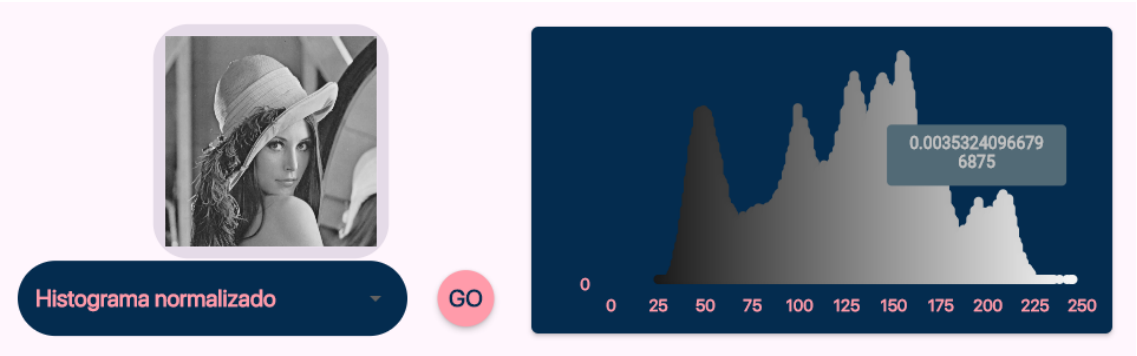


O cálculo do histograma de uma imagem é realizado através de uma varredura pixel a pixel da imagem.

```
Uint8List getLuminanceValues(Image decodedImage) {
  return decodedImage.getBytes(format: Format.luminance);}
```

2.2.2 Histograma normalizado

Um histograma normalizado é um histograma onde cada uma de suas posições é dividida pelo total de frequências de forma que a soma seja unitária. Essencialmente é um valor percentual para cada elemento ao invés de um valor absoluto, como o histograma básico.



O histograma normalizado é calculado através da normalização da frequência do histograma básico, onde cada valor de luminância é dividido pela quantidade de valores existentes:

```
final histogramData =
histogramGeneration(luminanceList).get<Map<int, num>>();
  final pixelCount = luminanceList.length;
  final normalizedFrequency = histogramData!.map(
    (key, value) => MapEntry(key, value / pixelCount),
  );
```

2.2.3 Enhancement

Procedimentos de “enhancement” são técnicas utilizadas para alcançar uma melhoria na qualidade de imagem em alguma aplicação específica, como remoção de ruídos, correções geométricas ou extração. Normalmente são procedimentos “ad hoc”, soluções específicas a um único tipo de problema ao invés de algo mais geral.

2.2.4 Equalização de histograma

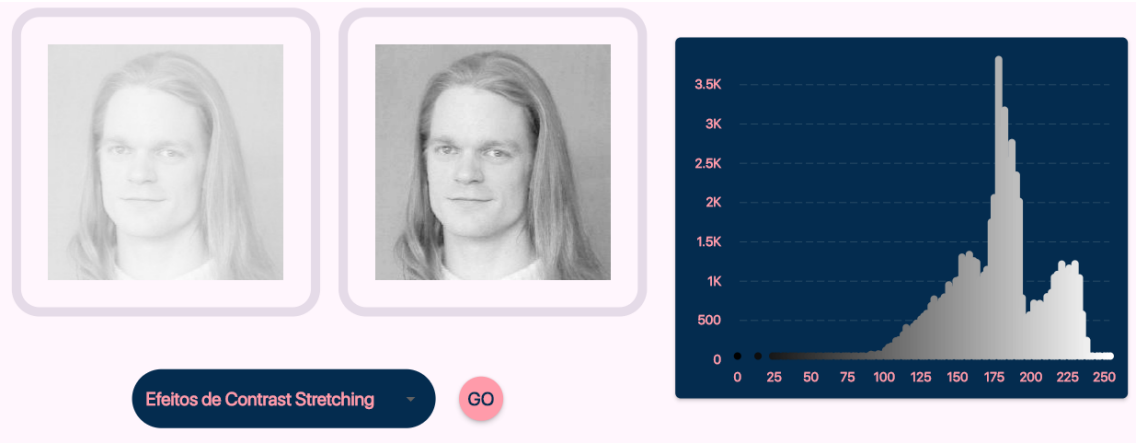
A equalização de um histograma é uma redistribuição de valores dos níveis de cinza em uma imagem na tentativa de obter-se um histograma uniforme.

É realizado em cada pixel individual da imagem, processando a luminância original de cada pixel e equalizando estes valores:

```
List<int> newImageLuminanceList = [];
  for (int pixelValue in originalImageLuminanceList) {
    newImageLuminanceList.add(newRk[pixelValue].toInt());}

  Map<int, num> newHistogram =
histogramGeneration(Uint8List.fromList(newImageLuminanceList))
          .get<Map<int, num>>()!;

  return Tuple(newHistogram,
Uint8List.fromList(newImageLuminanceList));
```

2.2.5 Efeitos de Contrast Stretching

Assim como o nome sugere, o algoritmo de Contrast Stretching estica os limites do contraste de uma imagem aos extremos. O menor valor de luminância é levado ao limite mínimo e o maior valor de luminância é levado ao limite máximo.



É realizado ao obter os extremos mínimo e máximo de luminância, mapeá-los a um intervalo {0; 255} e distribuir os valores restantes entre os extremos.

```
List<int> newImageLuminanceList = [];
  for (int pixelValue in originalImageLuminanceList) {
    newImageLuminanceList.add(newRk[pixelValue].toInt());}

  Map<int, num> newHistogram =
histogramGeneration(Uint8List.fromList(newImageLuminanceList))
          .get<Map<int, num>>()!;

  return Tuple(newHistogram,
Uint8List.fromList(newImageLuminanceList));
```

2.3 Conclusão

Estudar histogramas permite uma introspecção em como algoritmos de tratamento de imagens funcionam. Ajustes de softwares como Photoshop, Krita e Gimp começam a fazer sentido após um entendimento do que os gráficos e curvas da interface de ajustes de luminosidade, saturação e outros filtros significam.

Histogramas também são úteis em campos físicos, como a interpretação de imagens por um computador. A fotogrametria é um campo do processamento de imagens que utiliza um fluxo de imagens para interpretar objetos do mundo real em um espaço bi ou tridimensional.

3 Smoothing

3.1 Introdução

A suavização de imagens com filtros low-pass é um processo utilizado em tratamento de imagens para a remoção de ruídos de alta frequência. Cada pixel recebe um novo valor a partir da utilização de uma matriz de valores que avalia os valores de seus vizinhos, realiza um cálculo e retorna este novo valor a uma imagem de saída. É importante que a saída seja separada da imagem de entrada, visto que um pixel afeta o outro e uma mudança na entrada pode causar uma saída indesejada.

3.2 Filtro Espacial: Smoothing

A suavização de uma imagem através de filtros é uma operação espacial e de vizinhança (local) linear onde cada pixel é processado de acordo com seus vizinhos e seu resultado é passado a uma nova imagem. Funciona através da passada de uma matriz chamada de "máscara" por cada pixel. A matriz deve ser simétrica ao redor do pixel analisado, portanto, os tamanhos de máscara utilizados devem ser de ordem NxN, onde N é um valor ímpar maior ou igual a 3. Cada valor desta matriz de máscara é conhecido como "Peso".

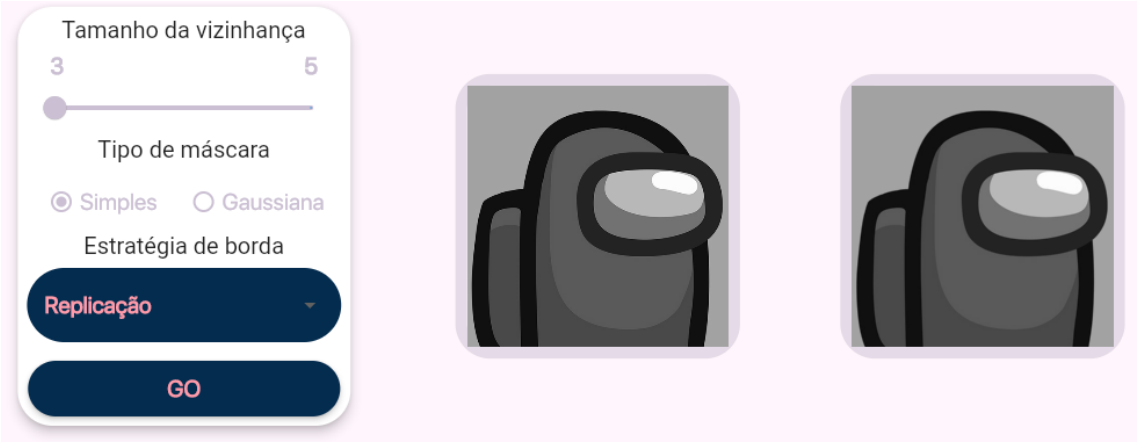
No cálculo de smoothing por média, o resultado de cada pixel vizinho ao central é multiplicado pelo seu valor correspondente na máscara e dividido pela soma dos valores da máscara.

```
List<int> getNeighborhood({
for (var y = yPosition - neighborhoodGroup;
      y <= yPosition + neighborhoodGroup;
      y++) {
    for (var x = xPosition - neighborhoodGroup;
        x <= xPosition + neighborhoodGroup;
        x++) {
      try { neighborhood.add(imageLuminanceMatrix[y][x]);
      } catch (_) {
        if (isConvolution) {
          final newY = y < 0 ? imageHeight + y : y %
imageHeight;
          final newX = x < 0 ? imageWidth + x : x % imageWidth;

          neighborhood.add(
            imageLuminanceMatrix[newY][newX],
          );
        } else {
          neighborhood.add(-1);}}}}

int applyMask(List<int> mask, List<int> neighborhood) {
  final maskSum = mask.reduce((a, b) => a + b);
  var pixelSum = 0;
  for (var position = 0; position < mask.length; position++) {
    pixelSum += mask[position] * neighborhood[position];}
  return pixelSum ~/ maskSum;
```

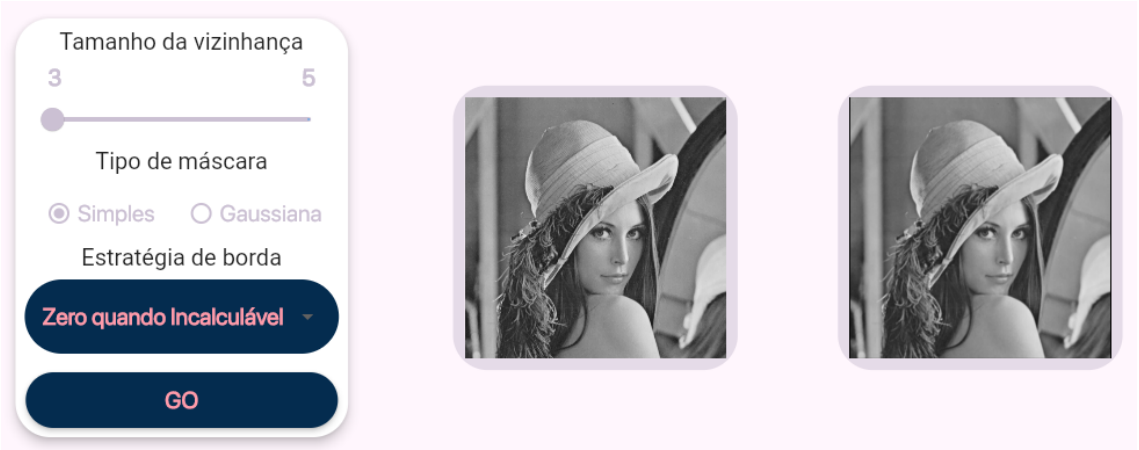
3.3 Estratégia de bordas

A natureza do preenchimento de uma matriz de saída pela utilização de uma máscara de média causa problemas ao calcular os valores nas extremidades, visto que os pixels destas não possuem todos os seus vizinhos para que o cálculo seja realizado. Para resolver este problema, algumas soluções foram propostas:

- **Replicação**: Os pixels das bordas da imagem original são replicados na imagem de saída. Preserva as informações da imagem original.



- **Zero quando incalculável**: Valores incalculáveis recebem zero.



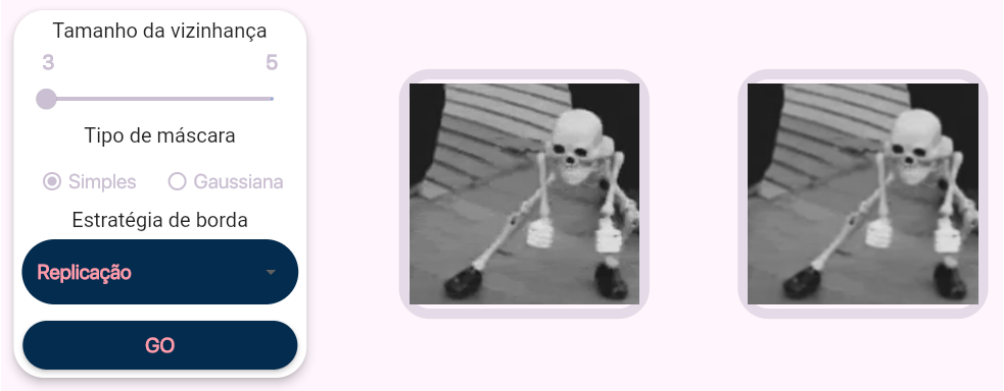
- **Padding com zeros**: Trata os valores inexistentes como zeros ao realizar o cálculo das bordas. Essencialmente cria uma borda com zeros ao redor da imagem original.

- **Convolução periódica**: A máscara é deslocada sobre todos os pixels da imagem original como se esta fosse adjacente em suas extremidades.

3.4 Máscaras

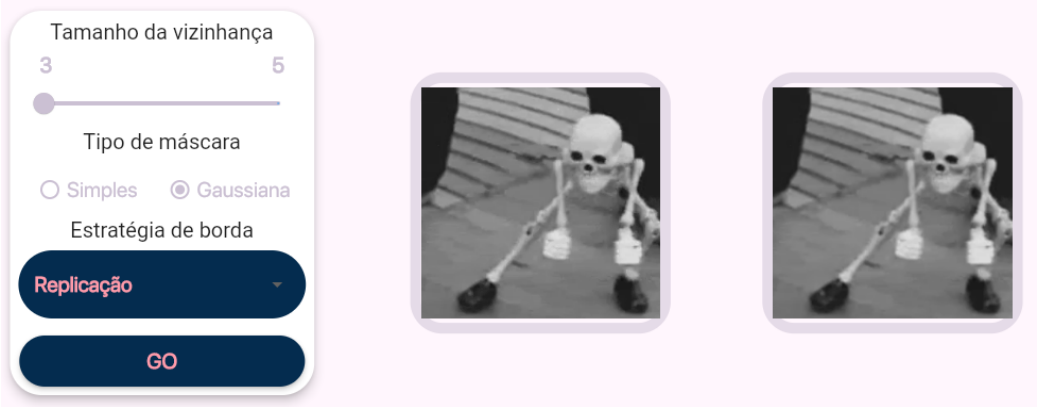
Máscaras são as matrizes utilizadas nas operações espaciais de vizinhança. Dividem-se em:

- Máscaras simples: Utilizam pesos positivos e seus tamanhos definem a magnitude do efeito



- Máscaras gaussianas: Pesos são amostras de uma função gaussiana dada por:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$



3.1 Conclusão

A principal utilidade de filtros de suavização como o borramento é a sua utilização no pré-processamento de imagens para remoção de ruídos, na extração de objetos maiores e na união de pequenos intervalos em linhas retas ou curvas. Sua utilização pode causar problemas ao tratar as extremidades de uma imagem, necessitando um cuidado no tratamento com diferentes possíveis abordagens. A máscara a ser utilizada pode variar a depender da aplicação em questão, visto que o processamento de diferentes imagens é uma atividade ad-hoc.

Filtros desta natureza também são úteis no pós-processamento de imagens para ocultar objetos no plano de fundo e na redução de aberrações cromáticas (pontos de alto contraste de diferentes cores ou seções).

4. Filtragem Espacial

Filtros espaciais são transformações de uma imagem numa base de pixels individuais que não dependem apenas do valor de cinza de um pixel individual, mas também de seus vizinhos. Utilizam-se de máscaras, matrizes cujo pixel referente é o centro de uma matriz de natureza NxN, onde N é obrigatoriamente um número ímpar. Têm por finalidade salientar determinados aspectos em imagens digitais ou reduzir ruídos.

4.1 Filtro Laplaciano

Um filtro Laplaciano provém da derivada da segunda ordem do valor da função que descreve a intensidade de seus pixels. Traz uma resposta mais acentuada a detalhes finos, como pontos isolados e linhas. O filtro Laplaciano é um filtro isotrópico, ou seja, a resposta é independente da direção da descontinuidade na imagem em que o filtro é aplicado.

```
List<int> neighborhood = parameter;
final product = neighborhood * laplaceMask;
return product.reduce(sum);
```

4.2 Filtro LoG

Laplaciano de Gaussiano é um algoritmo de detecção de bordas. Similar ao filtro Laplaciano, encontra a derivada de segunda ordem de uma imagem, retornando zero em áreas onde a imagem possui uma intensidade constante, um valor positivo quando a intensidade da vizinhança for mais escura e um valor positivo quando mais clara.

```
final laplace = -1 /
     (pi * pow(sigma, 4)) *
     (1 - (pow(x, 2) + pow(y, 2)) / (2 * pow(sigma, 2))) *
     exp(-(pow(x, 2) + pow(y, 2)) / (2 * pow(sigma, 2)));
 return laplace;
```

4.3 Sharpening, Unsharp Masking e Highboost Filtering

Sharpening refere-se a qualquer técnica de aprimoramento de imagens que realce arestas e detalhes finos numa imagem. É comumente utilizado em impressão e na indústria fotográfica para aumentar o contraste local em imagens.

Unsharp Masking é uma técnica cujo nome deriva do uso de uma imagem negativa e borrada para a criação de uma máscara da imagem original. Essa máscara é, então, combinada com o positivo original da imagem, criando uma versão menos borrada da original. Quando o coeficiente é igual a 1, o efeito resultante é conhecido como Unsharp Masking.

Highboost Filtering é uma técnica utilizada para enfatizar componentes de alta frequência sem que componentes de baixa frequência sejam afetados. É válido notar que quando o valor do coeficiente for igual a zero, o Highboost Filtering torna-se Sharpening Laplaciano padrão.

```
for (var y = 0; y < initialPixels.length; y++) {
        for (var x = 0; x < initialPixels[0].length; x++) {
          final neighborhood = getNeighborhood(
            imageLuminanceMatrix: imageLuminanceMatrix,
            yPosition: y,
            xPosition: x,
            neighborhoodSize: threeNeighborhood ? 3 : 9,
            isConvolution: true);
```

4.4 Gradiente

Operador comumente utilizado para a diferenciação de imagens. Composto por um vetor cuja direção indica os locais nos quais os níveis de cinza sofrem maior variação.

Pode ser calculado através das derivadas parciais:

$$\nabla f(x,y) = \frac{\partial f(x,y)}{\partial x}\, i + \frac{\partial f(x,y)}{\partial y}\, j$$

Onde I e J são vetores unitários nas direções X e Y, respectivamente.

4.5 Detector de Roberts / Sobel

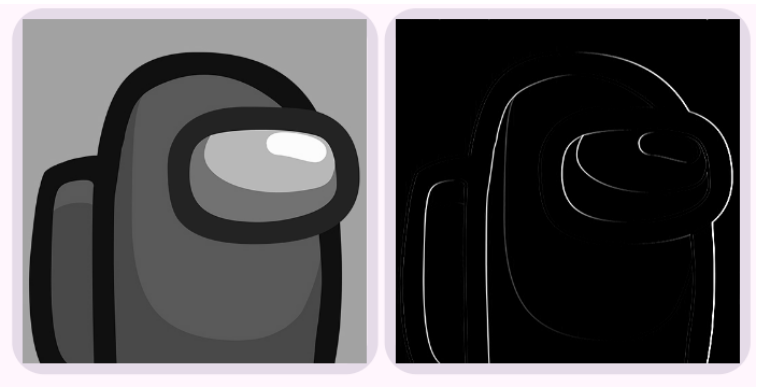
Os detectores de Roberts e Sobel são técnicas de detecção de bordas, úteis para encontrar contornos de objetos.

O detector de Roberts é encontrado através da primeira derivada da descontinuidade dos níveis de cinza para detecção de bordas. Duas máscaras são passadas pela matriz da imagem, uma esquerda e uma direita com os resultados de ambas sendo retornado a uma matriz resultante contendo as bordas da imagem original.

O detector de Sobel também é encontrado através da primeira derivada. Dada uma sequência de pixels, pode-se fazer uma analogia desta sequência com pontos de uma função. A derivada desta função indica pontos de mínimo e máximo, o que significa uma borda. Utiliza duas máscaras de convolução de Sobel, uma no eixo X e outra no eixo Y, com o resultado da passada de cada uma sendo colocado numa nova matriz.

A magnitude que será passada à nova matriz é calculada pela mesma fórmula de magnitude do detector de bordas de Roberts.

```
int robertsSobelFilter(dynamic parameter) {
  List<int> neighborhood = parameter;
  final product = neighborhood * _detectorMask!;
  return product.reduce(sum);}
for (var y = 0; y < initialPixels.length; y++) {
        for (var x = 0; x < initialPixels[0].length; x++) {
          final neighborhood = getNeighborhood(
            imageLuminanceMatrix: imageLuminanceMatrix,
            yPosition: y,
            xPosition: x,
            neighborhoodSize: threeNeighborhood ? 3 : 9,
            isConvolution: true);
```



4.6 Junção de vários filtros

4.7 Conclusão

A detecção de bordas é, em teoria, um problema simples de operações matriciais através da passagem de máscaras, mas é computacionalmente intensivo. Através das diferenças de luminosidade em cada pixel, é possível distinguir bordas e, por consequência, objetos. Por isso, são operações com aplicações nos mais variados campos, desde análise de imagens de satélite até a medicina. É a base para o campo de Machine Vision, particularmente nas áreas de detecção de características e extração de características.

A junção de vários filtros espaciais pode tanto aumentar a visibilidade de bordas como nulificar o efeito de filtros prévios. É algo que requer cuidado por parte do responsável pelo pré-processamento do banco de imagens.

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.

Ponti Jr, M. (2013) “Realce de imagens parte 1: operações pontuais”. Instituto de Ciências Matemáticas e de Computação - USP.

Backes, A. (2014) “Realce de imagens baseado em histogramas”. Faculdade de Computação / UFU - Universidade Federal de Uberlândia.

Lopes, R. (1999) “Image Enhancement”. Escola Politécnica de São Paulo - USP.

Dynamsoft (2019) “Image Processing 101 - Chapter 2.3: Spatial Filters (Convolution)”. Dynamsoft Corporation.

Agarwal, P. (2018) “Smoothing in Digital Image Processing”. JECRC Foundation.

Pound, M. (2015) “How Blurs & Filters Work”. Computerphile.

Jain, R., Katsuri, R. e Schunck, B. (1995) “Machine Vision”. McGraw-Hill, Inc.

Backes, A. (2014) “Filtragem espacial de imagem e convolução”. FACOM - Faculdade de Computação / UFU - Universidade Federal de Uberlândia.

NPTEL (2021) “Module 5.8: Discrete Laplacian Operators - Image Sharpening”. National Programme on Technology Enhanced Learning.

Fisher, R., Perkins, S., Walker, A., Wolfart, E. (2004). “Laplacian of Gaussian”. University of Edinburgh.

Alirezanejad, M., Saffari, V., Amirgholipour, S., Sharifi, A. M. (2014). “Effect of Locations of using High Boost Filtering on the Watermark Recovery in Spatial Domain Watermarking”. Indian Journal of Science and Technology, Vol 7.

Faria, F. A., Pedrini, H. (2015). “Processamento de Imagens”. UNIFESP.

Golts, G. A. M. (2011). “Redes Neurais Artificiais Em Imagens Para Estimação Da Posição De Um Vant”. INPE MTC.