Relatório dos Trabalhos T1 e T2
Processamento Digital de Imagens

Denilson Oliveira[1], **Gabriela Filete**[1], **Rafael Barbosa**[1], **Renato Souza**[1]

[1]Instituto de Ciências Exatas e Tecnológicas
Universidade Federal de Viçosa - *Campus* Rio Paranaíba (UFV-CRP)
Rodovia MG-230 – Km 7 – Rio Paranaíba-MG – Brasil
CEP: 38810-000 – Caixa Postal 22

{denilson.castro, gabriela.filete, rafael.domingos, renato.s.silva}@ufv.br

Abstract. *The purpose of this article is practical deepening of the discipline of Digital Image Processing (PDI). In it, concepts such as fundamentals of digital images, intensity transformations, spatial filtering for smoothing, spatial filtering for sharpening, segmentation for borders detection and segmentation for thresholding are discussed. Using the Python language, were made implementations in the form of program to manipulation the respective images.*

Resumo. *O presente artigo tem por finalidade o aprofundamento prático da disciplina de Processamento Digital de Imagens (PDI). Nele, são abordados conceitos como fundamentos de imagens digitais, transformações de intensidade, filtragem espacial para suavização, filtragem espacial para aguçamento, segmentação para detecção de bordas e segmentação para limiarização. Utilizando a linguagem Python foram feitas implementações na forma de programa para manipulação das respectivas imagens.*

1. Introdução

A área de processamento digital de imagens tem sido amplamente estudada afim de viabilizar aplicações para aprimorar informações pictóricas para interpretação humana e análise computadorizada de uma cena extraída [Filho and Neto 1999].

O processamento digital de imagens teve sua origem no ano de 1920, através do sistema de transmissão de imagens por cabo submarino Bartlane, onde era capaz de codificar imagens em cinco níveis de brilho distintos, chegando a quinze níveis nove anos depois [Gonzalez and Woods 2000].

O grande avanço na área veio com a aparição dos primeiros computadores de grande porte. Em 1964, um projeto espacial norte-americano utilizou técnicas computacionais nas imagens da lua transmitidas pela sonda Ranger, buscando corrigir as distorções pela câmera responsável pelas imagens [Filho and Neto 1999].

Após o avanço no ano de 1964, pesquisadores ao redor do mundo empenharam-se na busca de melhorias nos mais diversos ramos da atividade humana.

2. Fundamentos de Imagens Digitais

Matematicamente uma imagem é definida pela função *f(x,y)* da intensidade luminosa. Tal função representa o produto entre a quantidades de luz refletida sobre o objeto com as propriedade de transmitância própria do objeto [Filho and Neto 1999].

Para a aquisição de uma imagem é necessário fazer a conversão de um cenário tridimensional para analógico. Para isso deve-se reduzir a dimensionalidade, por meio de um dispositivo de aquisição de imagens. Após a aquisição da imagem é feito a discretização espacial e convertida a imagem analógica para digitalizada [Filho and Neto 1999].

Operações aritméticas e lógicas são utilizadas na manipulação de imagens. Abaixo é descrito cada operação, bem como testes para cada tipo de operação.

2.1. Operações Aritméticas sobre Imagens

Dentre as operações aritméticas, podem ser aplicadas a soma, subtração, multiplicação e divisão sobre as imagens. Essas operações são aplicadas pixel a pixel em cada pixel correspondente nas imagens a serem trabalhadas, gerando assim o novo pixel.

Para o trabalho em questão foi utilizado a operação aritmética de subtração. O algoritmo faz a leitura de duas imagens binárias diferentes, faz a subtração entre as duas imagens e gera a imagem processada de saída.

A Figura 1 mostra duas imagens de entrada e a respectiva imagem de saída, após feito a subtração.

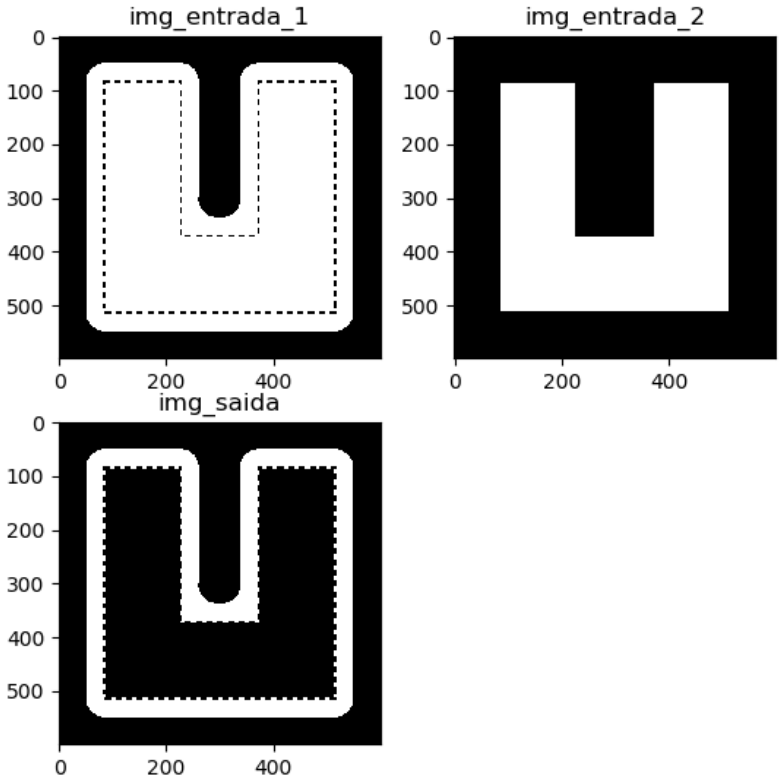

Figura 1. Aplicação da operação aritmética de subtração

Como observado na Figura 1, foram utilizadas duas imagens de entradas, img_entrada_1 e img_entrada_2, onde foi feito uma subtração de pixel por pixel entre as imagens. A

imagem de saída, img_saida, mostra que a parte tracejada interna da img_entrada_1 foi removida, passando de branco para preto.

2.2. Operações Lógicas sobre Imagens

Duas são as operações lógicas a serem aplicadas sobre as imagens. A operação AND, responsável por fazer a união das imagens, e a operação OR responsável pela intersecção entre as mesmas.

Nesse trabalho foi utilizada a operação lógica OR. O algoritmo faz a leitura de duas imagens, fazendo a disjunção das mesmas e gerando uma imagem de saída.

A Figura 2 mostra duas imagens de entrada e a intersecção entre elas como imagem de saída.

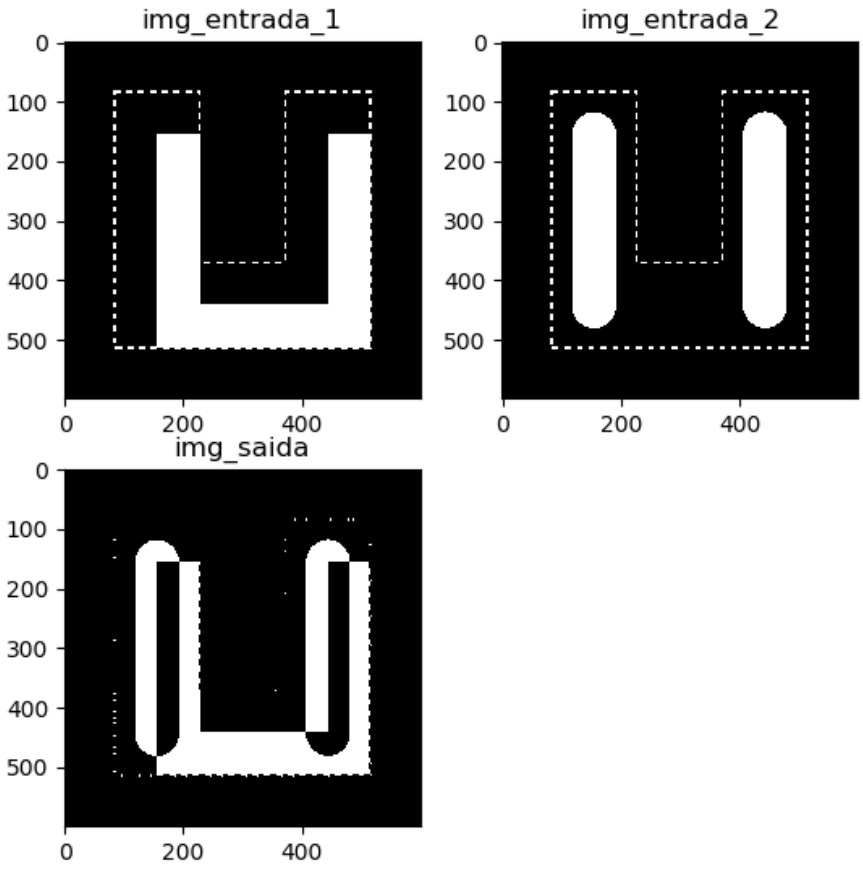

Figura 2. Aplicação da operação lógica OR

Observando a Figura 2, foram carregadas duas imagens de entradas, img_entrada_1 e img_entrada_2, onde foi feito a intersecção entre elas pela operação OR. A imagem de saída, img_saida, retrata o dito anteriormente, onde as partes em partes em comum foram removidas, passando de branco para preto.

3. Transformações de Intensidade

Na transformação de intensidade é feita uma filtragem linear com janela 1x1 e levando em consideração o valor de intensidade de cada pixel. Várias são os tipos para melhoramento de intensidades, cada qual com uma função específica. Abaixo são apresentados cinco delas.

3.1. Negativo de uma Imagem

O negativo de uma imagem binária nada mais é do que inverter os pixels brancos em escuros e os pixels escuros em brancos, tornando assim o inverso da imagem. Esse tipo de manipulação de imagem se faz mais útil nas situações onde a imagem original é muito escura. Comumente vemos este tipo de técnica em imagens médicas.

Para esta técnica é feito a leitura de uma imagem de entrada e após o processamento é feito multiplicando pixel a pixel por -1, invertendo os pixels e gerando como saída imagem negativa.

A Figura 3 mostra a imagem de entrada e seu negativo como imagem de saída.

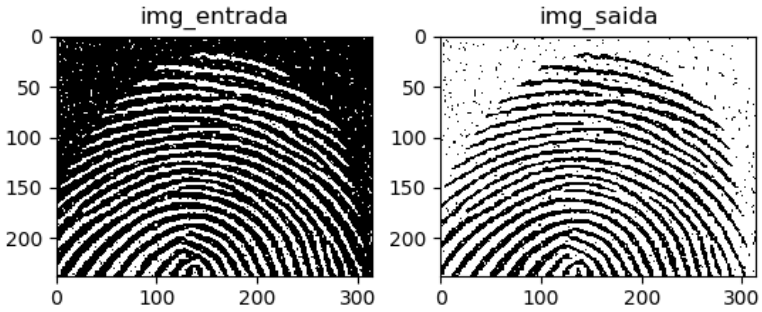

Figura 3. Negativo de uma imagem

Para a geração do negativo, foi feito o carregamento da img_entrada e a mesma sendo multiplicada por -1 pixel a pixel. Como se trata de uma imagem binária, pixels de cor branca foram transformados em preto e pixels de cor preta transformados em branco.

3.2. Transformação Gama

A transformação gama tem por função controlar o nível de brilho e contraste de uma imagem. Quando o valor de gama é menor 1 o contraste é maior e o brilho menor, quando a é igual a 1 não há alteração e quando é maior que 1 o contraste é menor e o brilho maior.

Para o processamento da imagem na linguagem Python para o método de transformação gama é aplicado uma função que utiliza a equação O = I**gamma escalando cada pixel no intervalo de 0 a 1. Os parâmetro utilizados nessa função são a img_entrada e o valor escolhido para o gama, afim de ajustar o brilho e contraste.

A Figura 4 mostra as imagens de entrada e a respectiva imagem de saída utilizando a transformação gama.

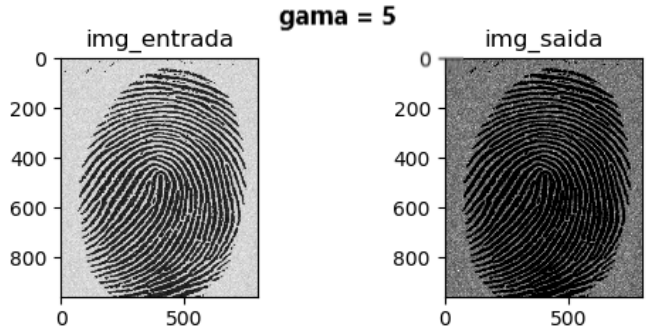

Figura 4. Transformação de potência (gama)

Para o processamento da Figura 4 foi utilizado um gama no valor 5, percebendo assim um escurecimento da imagem e melhor detalhamento dos pixels de tons mais claros.

3.3. Alargamento de Contraste

O contraste refere-se ao aumento da escala dos níveis de cinza, medindo a diferença de luminosidade entre as áreas claras e escuras. O realce pelo alargamento de contraste é considerado um processo mais simples, porém de grande eficácia.

Lida a img_entrada, para o alargamento do contraste da imagem, inicialmente foram definidos dois valores percentuais, de mínimo e máximo, para ajustar assim dentro desse limite. Após, a função responsável por esse alargamento entre em ação, pegando como parâmetro a imagem de entrada e os dois valores de mínimo e máximo. A resposta deste processamento se dá pela img_saida.

A Figura 5 apresenta a imagem lida, bem como a imagem gerada de saída.

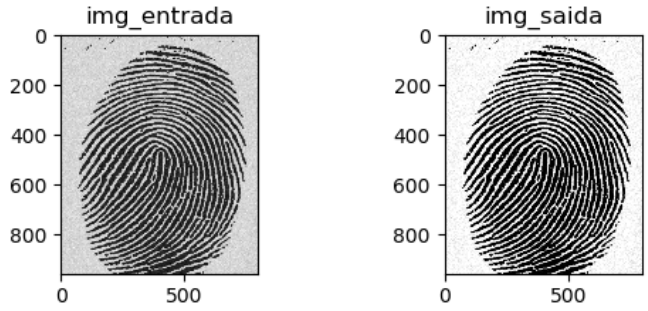

Figura 5. Alargamento de contraste

O resultado do processamento é exposto na Figura 5, onde percebe-se uma melhor distribuição dos pixels, tendo ganho de brilho e contraste.

3.4. Construção de Histogramas

A construção de histogramas se dá por uma função discreta onde descreve os níveis de intensidades de uma imagem. Tal descrição nos permite observar a quantidade de

ocorrências de cada intensidade, facilitando assim a análise das características de uma imagem.

No processo de construção do histograma foi feito o carregamento de uma imagem e sobre a mesma foi utilizada uma função *flatten()*, responsável por retornar a quantificação das intensidades presentes na imagem.

A Figura 6 mostra a imagem de entrada utilizada e o histograma da mesma, após execução do algoritmo.

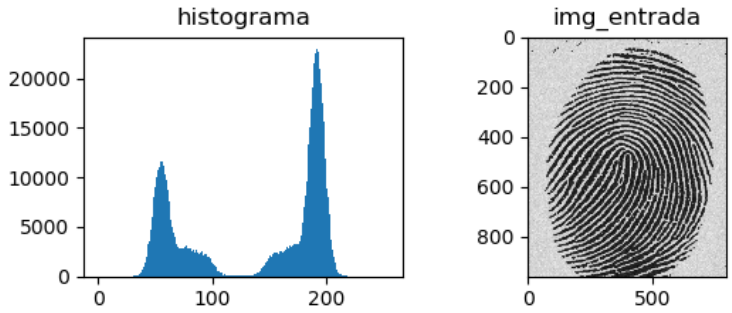

Figura 6. Construção de histograma

Observando o histograma gerado percebe-se a alta incidência de pixels de coloração escura, o que visualmente se percebe na imagem processada.

3.5. Equalização de Histograma

Na construção de histogramas citada anteriormente, percebe-se a concentração de faixas de intensidade em algumas regiões, deixando assim faixas com baixa ou nenhuma intensidade. O que a equalização de histograma busca fazer é distribuir uniformemente os níveis na curva de intensidades, para que o resultado de saída seja mais equilibrado.

Feita a leitura da imagem de entrada, o algoritmo aplica a função *exposure.equalize_hist()*, responsável por retornar a imagem de saída com níveis de intensidade mais distribuídos.

A Figura 7 mostra a imagem de entrada e sua respectiva imagem de saída.

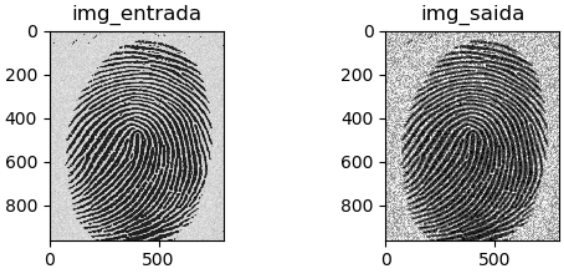

Figura 7. Equalização de histograma

Comparando a Figura 7 nota-se que a imagem de saída teve uma escurecida, isso se deu pelo fato da distribuição mais uniforme das intensidades.

4. Filtragem Espacial para Suavização

Na filtragem espacial para suavização são utilizadas máscaras, também chamadas de filtros espaciais. A suavização de imagens no domínio espacial faz uso de máscaras de convolução, geralmente usada para o borramento ou remoção de ruídos presentes na imagem [Filho and Neto 1999]. Algumas das técnicas para suavização são detalhadas abaixo.

4.1. Filtro de Média

O filtro da média é responsável por diminuir transições abruptas e suavizar os ruídos existentes numa imagem. Isso é feito substituindo cada pixel baseando-se na intensidade média de sua vizinhança.

No filtro da média é feito o carregamento da imagem de entrada e a mesma transformada como float, para possíveis divisões não inteiras. Na sequência, foi criado uma máscara de tamanho 5 e definida a operação para obtenção da média. Por fim, a função *filters.correlate()* realiza os cálculos e retorna a imagem de saída.

A Figura 8 mostra a imagem de entrada e a respectiva imagem de saída aplicado o filtro da média.

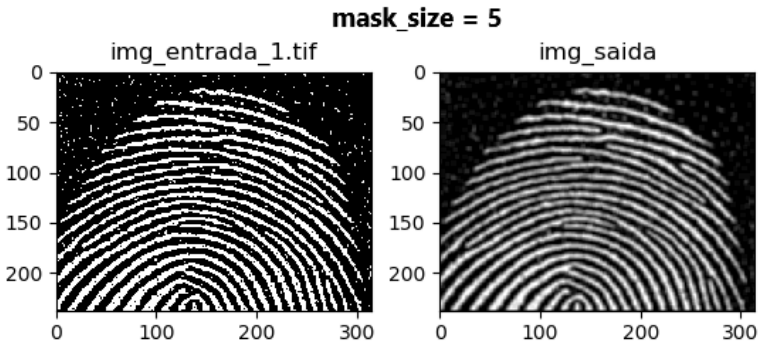

Figura 8. Filtro da média

A imagem de saída nos mostra uma imagem suavizada da imagem de entrada, onde foram reduzidos os ruídos existentes nela.

4.2. Filtro Gaussiano

O filtro gaussiano utiliza mais de um cálculo para suavização da imagem. Faz uso da curva de distribuição normal padrão, cálculo da média e desvio padrão para calcular a curva de Gauss.

Apesar de aparentemente parecer possuir maior complexidade na implementação por utilizar três cálculos para obtenção do filtro gaussiano, a linguagem Python facilitou tal ação, sendo necessário uma única função, *filters.gaussian_filter()*, passando como parâmetro o sigma com valor 15.

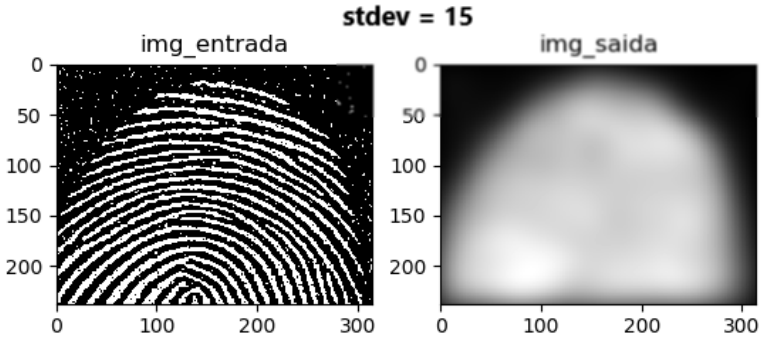

Figura 9. Filtro gaussiano

A Figura 9 mostra as imagem de entrada e a imagem de saída após aplicada a função gaussiana.

Comparando com a técnica de filtro da média, nos mostra que ambas suavizam a imagem, porém o filtro gaussiano retorna maior borramento na aplicação da técnica.

4.3. Filtro da Mediana

A técnica de filtro da média, diferentemente do filtro gaussiano visa reduzir o borramento acentuado da imagem preservando as bordas na suavização. É definido uma máscara e para cada pixel da imagem calculado o valor mediano de sua vizinhança.

Como na técnica de filtro da média, é feito o carregamento da imagem de entrada e após transformada como float. Obtida essa transformação a função *filters.median_filter()* calcula o valor central e altera o pixel. Para a análise em questão foi utilizado o valor 5 como tamanho da máscara.

A Figura 10 mostra as imagem de entrada e a juntamente com a imagem de saída.

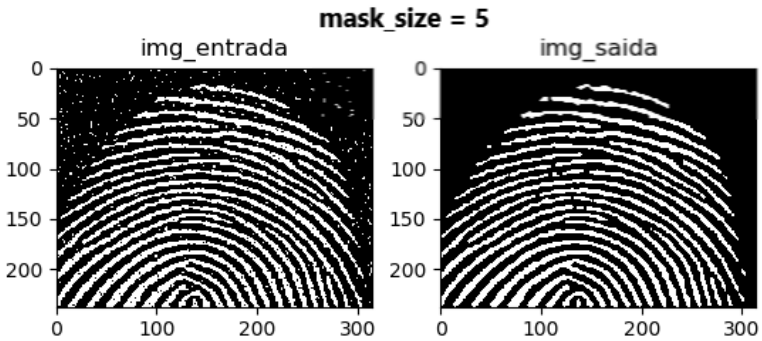

Figura 10. Filtro da mediana

Verifica-se que pixels de valores mais diferentes dentro da máscara foram alterados, obtendo assim uma imagem sem grandes ruídos.

4.4. Filtros de Máximo e Mínimo

Como o próprio nome diz, os filtros de máximo e mínimo dão ênfase aos valores máximos e mínimos a máscara será aplicada, onde o valor de cada pixel é substituído baseando-se nos pixels de sua vizinhança.

Para esta técnica, a imagem novamente é transformada como float e após aplicado as funções *filters.maximum_filter()* e *filters.minimum_filter()* para máximo e mínimo, respectivamente. O tamanho da máscara foi definido como 3, gerando duas imagens de saída.

A Figura 11 mostra a imagem de entrada e as duas imagens de saída, uma para máximo e outra para mínimo.

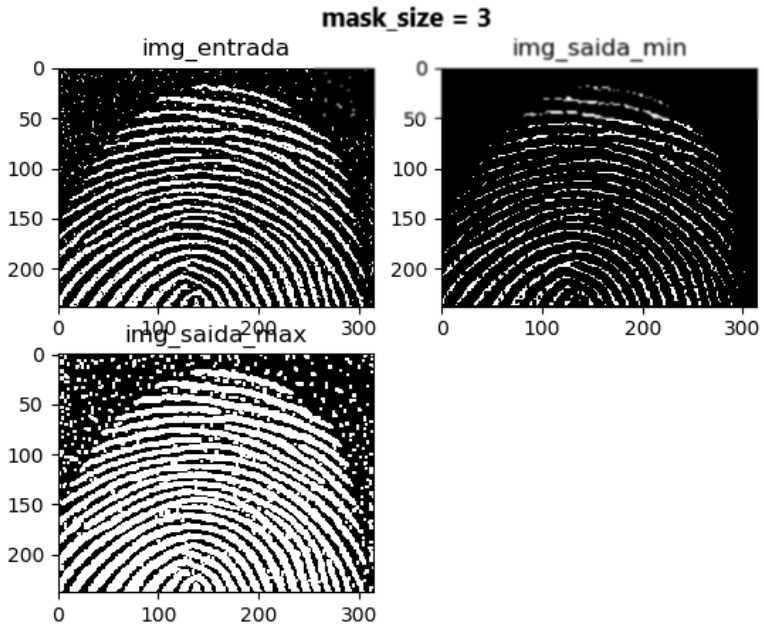

Figura 11. Filtros de máximo e mínimo

Com base nas figuras de saída alcançadas, observa-se que para os filtros de máximo e mínimo, uma clarea enquanto a outra escurece a imagem.

5. Filtragem Espacial para Aguçamento

Filtros espaciais para aguçamento buscam enfatizar pequenos detalhes realçando assim detalhes borrados [Gonzalez and Woods 2000]. Desta forma, destaca transições de intensidade entre diferentes elementos na imagem, aumentando assim a nitidez. O aguçamento é calculado através da diferenciação no domínio do espaço.

Ainda segundo O uso desta técnica está nos diversos seguimentos, desde uma simples impressão eletrônica a detecção de alvos em armas inteligentes [Gonzalez and Woods 2000].

5.1. Laplaciano

O operador derivativo isotrópico mais simples é o laplaciano. Filtros isotrópicos são invariantes em rotação, isso significa que rotacionar a imagem e depois aplicar o filtro dá o mesmo resultado que aplicar o filtro e depois rotacionar a imagem.

Nesta técnica, a imagem é transformada para float. A metodologia consiste em criar uma máscara de filtragem a partir de uma fórmula discreta da derivada de segunda ordem definida. Dessa forma, o laplaciano é um operador linear, porque todas as derivadas são operações lineares. Após a máscara estar definida, é aplicada a imagem pela função *filters.correlate*. Por fim, é retornado a imagem de saída.

A utilização do laplaciano realça as descontinuidades de intensidade e atenua as regiões com níveis de intensidade de variação mais suave. Isso tende a produzir imagens em que linhas de borda e outras descontinuidades aparecem em tons de cinza sobrepostos a um fundo preto. Para resolver esse problema, soma-se a imagem laplaciana à original, conservando o efeito de aguçamento do laplaciano enquanto ”recupera-se”as características do fundo.

A Figura 12 mostra a imagem de entrada e a respectiva imagem de saída.

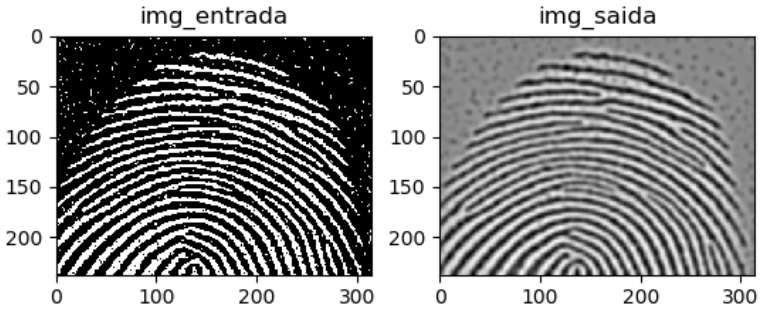

Figura 12. Laplaciano

É possível ver na imagem de saída as bordas e descontinuidades em tons de fundo. Essa imagem, somada à original que produz o aguçamento.

5.2. Máscara de Nitidez e *High-Boost*

Essa técnica realiza o aguçamento, ou aumenta a nitidez, de imagens subtraindo uma versão suavizada de uma imagem da sua versão original.

O primeiro passo desta técnica, consiste no borramento da imagem original, tal processo é realiado utilizando a função *ndimage.gaussian_filter* passando a imagem de entrada e valor de sigma sendo 7, como parametro. Após a obtenção da nova imagem borrada, é realizado a subtração da imagem original por esta, resultando em uma máscara que será adicionada a imagem original. Antes de se realizar o processo de soma das imagens, é multiplicado a máscara por um determinado K, sendo utilizado o valor de 4,5 para este, realizando a filtragem *High-Boost*.

A Figura 13 mostra a imagem de entrada e a respectiva imagem de saída.

É possível perceber na imagem de saída a versão ”suavizada”da imagem original. Aplicando os passos explicados acima, é gerada uma máscara que será somada à original, e essa máscara é a que podemos ver na imagem.

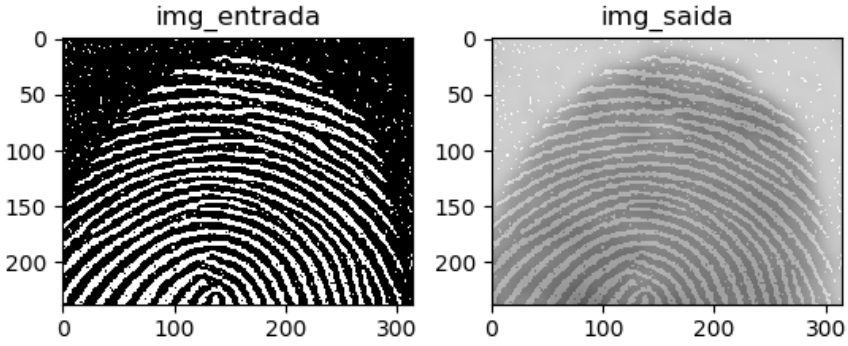

Figura 13. Máscara de nitidez e *high-boost*

5.3. Gradiente

O gradiente é a ferramente ideal para encontrar a intensidade e a direção da borda de uma imagem. Para obter os componentes de gradiente em cada endereço de pixel da imagem são usadas máscaras. As máscaras de Sobel são preferíveis por apresentarem melhor supressão de ruído, que é uma questão importante quando se lida com derivadas.

Para o gradiente, é realizada, após a imagem ser carregada, a transformação para float, para possíveis divisões não inteiras. Logo após, é definido a mascara de tamanho 5x5, utilizando-se os operadores de Sobel. São realizados os cálculos, pela função *filters.correlate()*, sendo seus parâmetros, a imagem de entrada e a máscara. Por fim, retorna a imagem de saída.

A Figura 14 mostra a imagem de entrada e a respectiva imagem de saída.

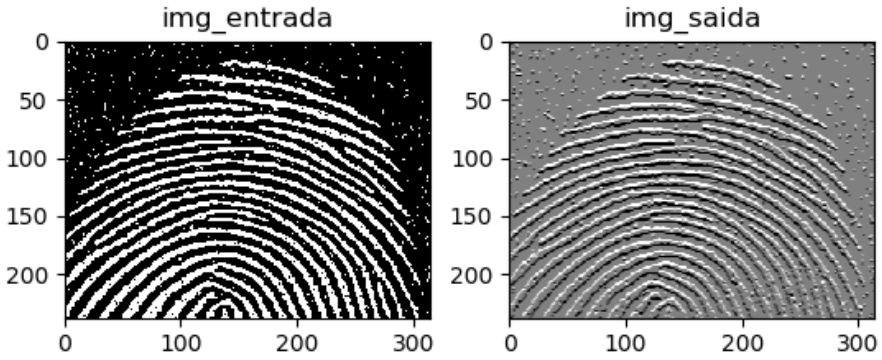

Figura 14. Gradiente

Com o resultado da imagem de saída obtido para a realização dos operadores de Sobel na horizontal, pode ser observado que os objetos definidos mais horizontalmente foram bem segmentados enquanto nem tanto os objetos mais a vertical na imagem.

6. Segmentação - Detecção de Bordas

A segmentação, se tratando da detecção de bordas, tem por objetivo distinguir pontos em uma imagem que há uma transição abrupta na intensidade luminosa. Essas alterações

repentinas na maioria dos casos representam ocorrências importantes.

6.1. Efeitos da Suavização da Detecção de Bordas

A suavização tem por objetivo eliminar possíveis ruídos que aparecerão como pequenas bordas. É um processo seletivo onde regiões uniformes são suavizadas conforme evolução aplicada.

O processo de Suavização da detecção de bordas é iniciado, transformando a imagem carregada em float. Após, é aplicado o processo do filtro da média com máscara obtida por parâmetro de entrada, para se obter a suavização da imagem. Na próxima etapa, realiza-se o gradiente com os operadores de Sobel em X e Y. Ao fim, é obtido a imagem de Magnitude do gradiente e retorna a imagem de saída.

A Figura 15 mostra a imagem de entrada e a respectiva imagem de saída.

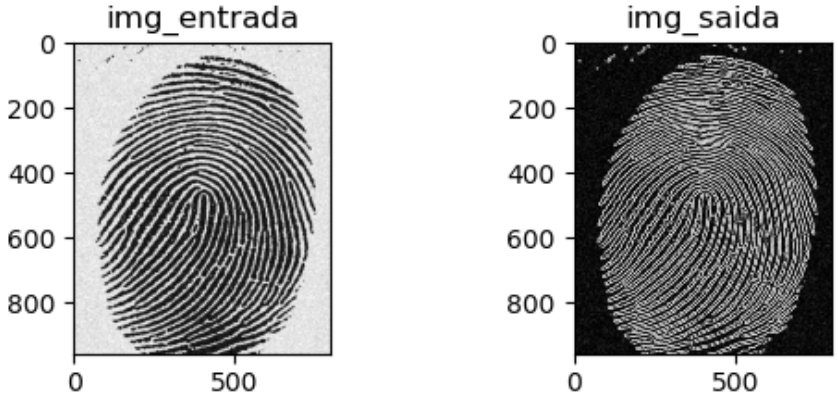

Figura 15. Efeitos na suavização na detecção de bordas

Na imagem resultante do processo, pode-se observar que foram segmentados as bordas dos objetos maiores e também de alguns ruídos, tendo alguns eliminados durante o processo de suavização.

6.2. Efeitos da Suavização e Limiarização na Detecção de Bordas

Suavizando a imagem busca-se remover ruídos na imagem deixando-a assim mais uniforme. Após feita a suavização aplica a limiarização para encontrar bordas mais fortes para melhor separá-las do restante da imagem.

Neste processo, os passos realizados são os mesmos do processo anterior, com adição de um Limiar sendo aplicado a toda a imagem. Tal processo é realizado aplicando a segmentação por um limiar definido previamente como sendo 20% da maior intensidade da imagem. Por fim, retorna a imagem de saída.

A Figura 16 mostra a imagem de entrada e a respectiva imagem de saída.

Com o acréscimo do processo de limiarização, pode-se observar bordas mais bem definidas e a eliminação dos ruídos de fundo, restando apenas alguns maiores.

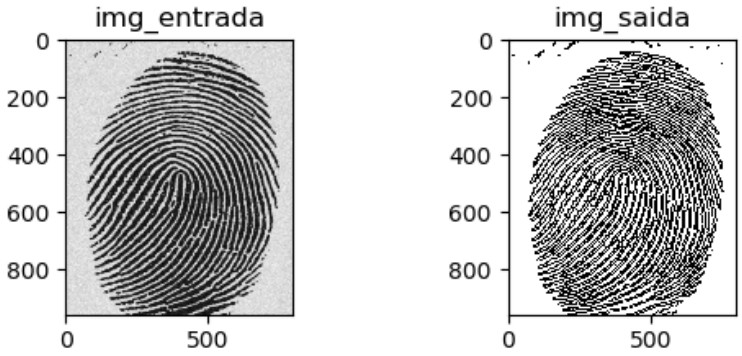

Figura 16. Efeitos da suavização e limiarização na detecção de bordas

7. Segmentação - Limiarização

A segmentação, se tratando da limiarização, se baseia na diferença dos níveis de cinza presentes nos objetos de uma imagem. Seu uso se dá pela definição de um limiar, de acordo com as características que se deseja isolar.

7.1. Limiarização Iterativa

A limiarização iterativa, como o próprio nome sugere, é utilizado aplicando sucessivas varreduras sobre uma imagem, objetivando a não utilização do seu histograma. A ideia é a cada iteração melhorar o valor do limiar.

O processo se inicia, transformando a imagem carregada em float. O próximo passo é encontrar o limiar, que seja satisfaça a condição de ser valor menor que um valor definido previamente, ou seja, encontrar um Ti em que a variação com o Ti-1 seja menor que o valor definido, é realizado separando pixels de fundo e de frente, encontrando suas médias e se calcula um novo limiar dividindo a soma das médias dos dois grupos, passos sucessivos são realizados até que a condição se satisfaça. Por fim é aplicado a segmentação utilizando o ultimo limiar encontrado e se retorna a imagem de saída.

A Figura 17 mostra a imagem de entrada e a respectiva imagem de saída.

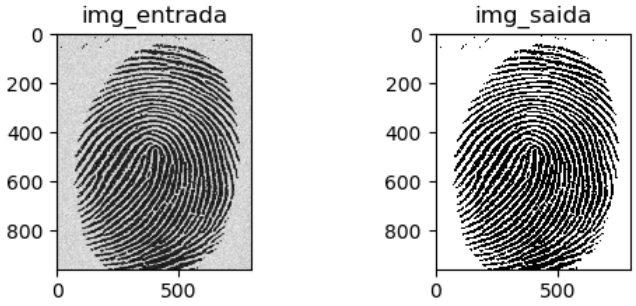

Figura 17. Limiarização iterativa

Como resultado, foi obtido uma imagem com os objetos segmentados, restando alguns ruídos maiores e com a eliminação dos pixeis cinza do fundo.

7.2. Limiarização Utilizando o Método de Otsu

O método de Otsu tem como objetivo obter os elementos do fundo e da frente de uma imagem em tons de cinza. Os elementos são separados pelo *threshold*, que deve ser determinado um valor ideal de forma que de todos os possíveis valores seja escolhido o que valoriza a soma da variância intraclasses da imagem.

O processo se inicia, transformando a imagem carregada em float. O próximo passo é encontrar o limiar de Otsu, sendo encontrado, passando a imagem pela função *filters.threshold_otsu*. Por fim é aplicado a segmentação utilizando o limiar encontrado e se retorna a imagem de saída.

A Figura 18 mostra a imagem de entrada e a respectiva imagem de saída.

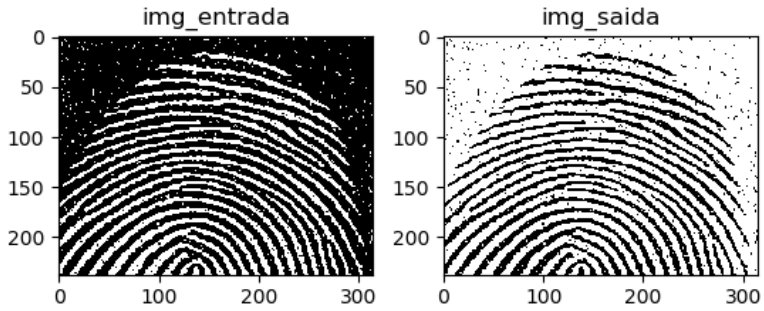

Figura 18. Limiarização utilizando o método de Otsu

Como pode ser percebido pela imagem, o algoritmo obteve excelente resultado por se tratar de um histograma bimodal.

7.3. Efeito da Suavização na Limiarização

A suavização na limiarização é usada quando o ruído não pode ser reduzido na fonte e o método de segmentação escolhido é a limiarização. O ruído pode transformar um problema simples de limiarização em um problema sem solução. Ao suavizar a imagem e só depois aplicar a limiarização, o desempenho melhora muito. Porém a indefinição da fronteira pode causar ligeira distorção entre o objeto e o fundo da imagem. Quanto mais agressiva a suavização da imagem, mais erros nas fronteiras acontecerão.

Ao iniciar, a imagem carregada é convertida para valores float. Próximo passo consiste em realizar o processo de filtro da Média, explicado anteriormente. E por fim, a segmentação pelo método de Otsu, do processo anterior. Por fim, é retornado a imagem de saída.

A Figura 19 mostra a imagem de entrada e a respectiva imagem de saída.

A imagem de saída é uma imagem bem limpa, se compara a imagem original. Se fosse aplicada apenas a limiarização, ser verificarmos na imagem obtida no método aplicado acima, é possível perceber ruídos, vários pontinhos no fundo. Já nessa imagem que foi suavizada antes da limiarização, o resultado que obtemos é um fundo limpo.

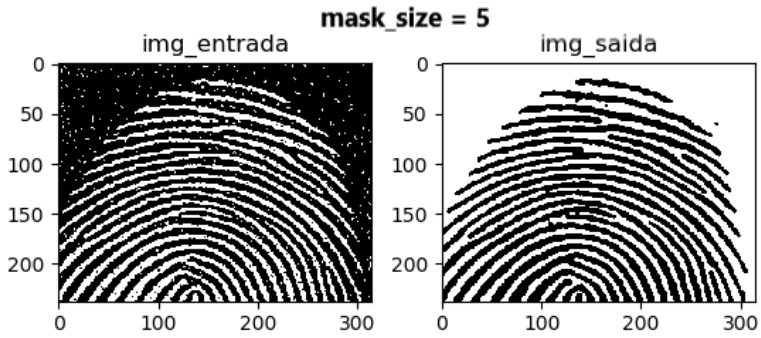

Figura 19. Efeito de suavização na limiarização

8. Conclusões

O presente trabalho teve como foco um melhor aprofundamento dos métodos de processamento de imagens que foram estudados ao longo da disciplina.

Com as implementações feitas e as imagens aplicadas a elas, foram feitos testes e observado as mudanças que cada técnica resulta numa imagem.

Contudo, percebe-se a grande utilidade do processamento digital de imagens, visto que as mesmas podem ser geradas à partir de diferentes resoluções sendo necessário a melhora de acordo com sua finalidade.

Referências

Filho, O. M. and Neto, H. V. (1999). *Processamento Digital de Imagens*. Brasport.

Gonzalez, R. C. and Woods, R. E. (2000). *Processamento de Imagens Digitais*. Edgard Blucher.