Ednilson Cardoso dos Santos
Eduardo Fillipe da Silva Reis
Thiago Joaquim Lima Ferreira

Relatório de implementação e casos de uso da ImageX

Aracaju, Sergipe

Junho, 2021

Ednilson Cardoso dos Santos
Eduardo Fillipe da Silva Reis
Thiago Joaquim Lima Ferreira

Relatório de implementação e casos de uso da ImageX

Relatório entregue como requisito de avaliação da primeira unidade da disciplina Processamento de Images, sob orientação da Prof. Dra. Beatriz Oliveira, com o objetivo de descrever a solução e implementação do problema de Reconhecimento Ótico de Caracteres, utilizando-se de técnicas de processamento e extração de características em imagens.

Universidade Federal de Sergipe - Campus São Cristóvão

Departamento de Computação

Programa de Graduação

Aracaju, Sergipe

Junho, 2021

Resumo

Este relatório introduz e analisa o software ***ImageX (imgX)***. Uma biblioteca construída utilizando a linguagem de programação Python e que possui como base o framework de computação científica Numpy. Inicialmente é apresentada uma introdução que visa mostrar as bases teóricas nas quais se apoiou a modelagem solução do projeto. Após isso, é discutida a solução do problema de OCR, através da biblioteca ImageX, desde sua modelagem teórica até detalhes de implementação. Por fim, são expostas algumas considerações finais que explanam os resultados obtidos e quais as próximas etapas do projeto.

Palavras-chaves: ImageX, Reconhecimento Ótico de Caracteres, OCR, Processamento de Imagens.

Lista de ilustrações

Sumário

Parte I

Fundamentação Teórica e Objetivos

1 Fundamentação Teórica

1.1 Introdução

A fotografia surgiu ainda na década de 1830, criada pelos dois irmãos Niépce e Daguerre, ao longo dos tempos ela foi se tornando cada vez mais conhecida e tomando diversos significados (MAUAD, 1996). Posteriormente, em 1957 foi criada a primeira imagem digital por Russell Kirsch (KIRSCH et al., 1957). Hoje em dia podemos definir uma imagem como uma função, f(x,y) em que x e y representam pontos no plano (GONZALEZ; WOODS, 2006).

O termo *Processamento de imagens digitais* se refere ao processamento de imagens por computadores. Não existe um acordo sobre em que exatamente a área atua, podendo "invadir"outras áreas de estudos como visão computacional e análise de figuras. Porém, podemos definir a área a partir do conceito que o processamento deve receber uma imagem e entregar outra como resultado (GONZALEZ; WOODS, 2006).

Atualmente a área de processamento digitais é extremante importante para diversas áreas de estudos e de aplicação, uma delas é a área médica, como no caso citado por Escarpinati (2016), que usa de imagens digitais para na análise de mamografias; também vemos o uso desta tecnologia em outras áreas, como até mesmo na determinação de volumes de pilhas de madeira (SILVA, 2003).

1.1.1 Reconhecimento Ótico de Caracteres (OCR)

Assim como as áreas citadas acima, o reconhecimento ótico de caracteres é mais uma área de processamento de imagem que está bastante presente no nosso dia a dia, por exemplo, no reconhecimento automático de placas de trânsito (BIANCHI, 1999). Para que softwares deste tipo funcionem muitas vezes são utilizadas técnicas de visão computacional e inteligência Computacional para o fazer reconhecimento de padrões.

Para complementar Gonzalez e Woods (2006) explica, que um sistema de visão computacional possui as seguintes etapas: Aquisição de imagens, pré-processamento, segmentação, extração dos atributos, identificação dos padrões previamente estabelecidos e por fim o relatório com análise e informações.

1.1.2 Trabalho realizado

Com essas informações é notória a importância do uso de processamento de imagens e suas técnicas no dia a dia das pessoas, sendo de vital importância para a computação

sociedade em geral.

Desta forma, foi proposta a criação de um software que se idealiza na forma de uma biblioteca voltada para o processamento de imagens, a ImageX. Neste software estão contidos algorítimos e técnicas voltadas ao processamento de imagens e que podem ser utilizadas de forma geral, inclusive para o problema exposto nesse relatório, o reconhecimento ótico de caracteres, aplicando diferentes conceitos aprendidos durante a disciplina para a realização do objetivo requisitado.

2 Objetivos

Neste capítulo serão apresentados os objetivos, gerais e específicos do trabalho realizado.

2.1 Geral

Desenvolver uma biblioteca para a realização de processamento de imagens, assim como o estudo de das técnicas pedidas.

2.2 Específicos

Para alcançar o objetivo geral, os seguintes objetivos específicos foram realizados:

- Analisar os requisitos apresentados para o programa pela professora;
- Desenvolver um algorítimo usando a técnicas desenvolvidas em sala de aula;
- Desenvolver a arquitetura da aplicação;
- Testar e analisar a ferramenta;
- Validar a ferramenta.

Parte II

Ferramentas Utilizadas

3 Ferramentas Utilizadas

3.1 O Framework NumPy

O NumPy é um pacote fundamental e largamente utilizado em computação científica, em conjunto com a linguagem de programação Python. Isso se deve à dois principais fatores. O primeiro refere-se à facilidade de uso que o NumPy proporciona, visto que faz uso as api Python de alto nível para realizar operações matemáticas e matriciais complexas em poucos comandos, tornando o desenvolvimento de aplicações algebricamente mais naturais. O segundo se trata da alta eficiência desse pacote, que se utiliza de algoritmos altamente otimizados (NUMPY, 2021), que permitem o uso de vetorização, paralelismo e outros recursos que tornam a execução de aplicações até 70 vezes mais rápidas. (NUMPY, 2021)

O NumPy é a base de um dos produtos desenvolvidos durante esta disciplina (abordado na próxima seção), o ImageX, que usa as estruturas de dados e algoritmos matriciais do NumPy nos algoritmos de processamento de Imagem.

3.2 ImageX

A ImageX(imgX) é uma biblioteca OpenSource sob a licença GPL v3. No vocabulário *pythônico*, um pacote, capaz de realizar o processamento de imagens em geral, aplicando máscaras, filtros, transformações e outros tipos de operações. O ImageX é uma implementação prática da teoria vista em sala de aula e no livro *"Digital Image Processing"* de Gonzales e Woods durante a disciplina Processamento de Imagens, no ano de 2021.

A ImageX foi desenvolvido utilizando a versão 3.9, mais recente da linguagem Python. Não está disponível em repositórios públicos do Python, como o PIP. Mas pode ser localizada no repositório oficial da biblioteca no GitHub: <https://github.com/eduardo-fillipe/imgx-framework>. Após o download, uma vez no diretório principal da biblioteca, pode ser instalada, como qualquer outro pacote, num ambiente virtual python através no comando: *python setup.py install*, e então ser utilizada em outros projetos de forma independente.

3.2.1 Suporte oferecido

As próximas seções descrevem o suporte oferecido pela biblioteca e seu funcionamento em sua versão atual, 1.2.1.

3.2.1.1 Quanto aos formatos de imagem

Atualmente, a imgx oferece suporte à leitura e escrita de imagens no formato *Netpbm* e suas variantes descritas no formato ASCII:

- PBM: *Portable BitMap*
- PGM: *Portable GrayMap*
- PPM: *Portable PixMap*

3.2.1.2 Quanto aos tipos de imagem

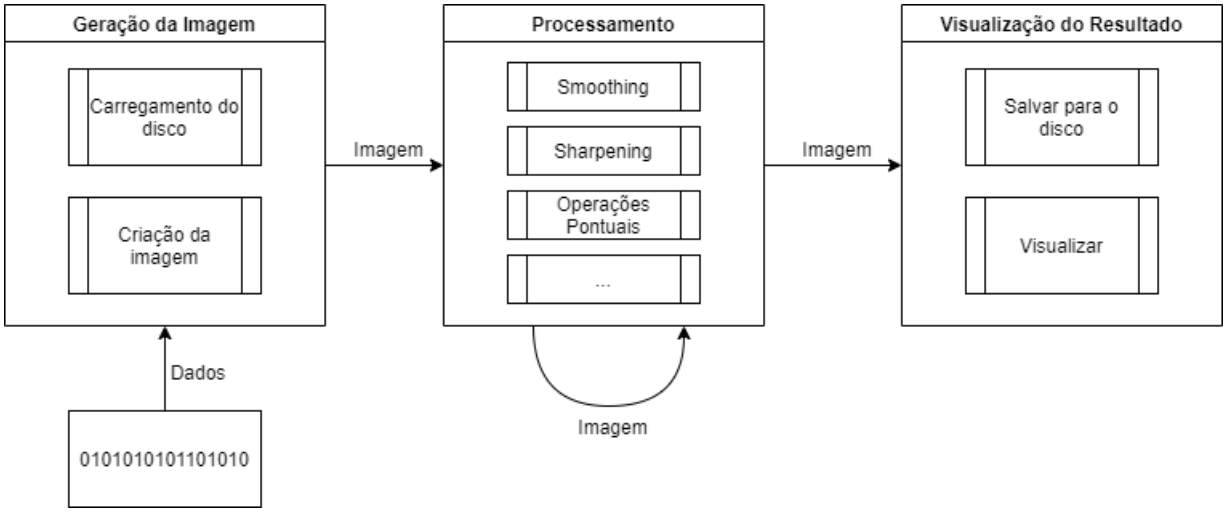
Todas as operações implementadas na biblioteca possuem suporte para os seguintes tipos de imagem.

- Coloridas: Tanto RGB quanto RGBA.
- Tons de cinza
- Binárias

3.2.2 Arquitetura

Todo o fluxo de processamento de uma imagem dentro da ImageX começa na leitura ou criação de uma imagem. Após, o processamento, onde algum ou alguns algoritmos podem ser aplicados sobre a imagem de entrada para gerar uma imagem resultante. Por fim, o resultado pode ser exibido e/ou salvo em disco. O processo pode ser visualizado no diagrama abaixo.

Figura 1 – Fluxo principal da ImageX



Todos os processos elencados são iniciados e finalizados dentro da própria biblioteca, sem a necessidade da utilização de pacotes externos, que possui as funções e métodos necessários para isso. Tais funções são organizadas em 4 sub-pacotes que concentram funções relacionadas, listados abaixo.

- **io**: Reúne as funções relacionadas à entrada e saída de dados, utilizadas para serializar e desserializar as estruturas de dados da imgX.
- **mask**: Pacote que engloba a implementação das máscaras de convolução utilizadas na biblioteca.
- **types**: Reúne as estruturas de dados utilizadas na ImageX. Em especial, a classe *Image*, que funciona armazena os dados da imagem na forma de um *NumPy ndarray* e demais meta atributos da imagem.
- **util**: Pacote para funções utilitárias da biblioteca.

3.2.3 Estruturas de Dados

Como introduzido na seção anterior, o tipo *Image* é a principal estrutura de dados da biblioteca e realiza a comunicação entre todos os módulos da mesma, além de criar uma interface de alto nível entre os algoritmos de processamento de imagens e o usuário consumidor.

A classe *Image* funciona como um contêiner que armazena todos os dados e meta-informações que uma imagem possa possuir: desde os valores de cada pixel, até comentários deixados nos cabeçalhos de cada protocolo de descrição como mostrado na figura 2.

Como se pode perceber na figura 2, os dados da imagem são armazenados na forma de um NumPy array. A depender do tipo de imagem (RGBA, RGB, Tons de Cinza ou preto e branca) o número de dimensões desse array varia da seguinte forma: Onde **M** é o número de linhas e **N** é o número de colunas do mesmo.

- Imagens RGBA: M x N x 4
- Imagens RGB: M x N x 3
- Imagens em Tons de cinza: M x N
- Imagens Binárias: M x N

Ainda, na figura 2, é importante salientar que o retorno de cada método, quando aplicável, é também um tipo *Image*. Isso se deve ao comportamento padrão fundamental da biblioteca em funções que realizam alterações do estado de um objeto *Image*. Assim, um

Figura 2 – Representação em alto nível da classe Image.

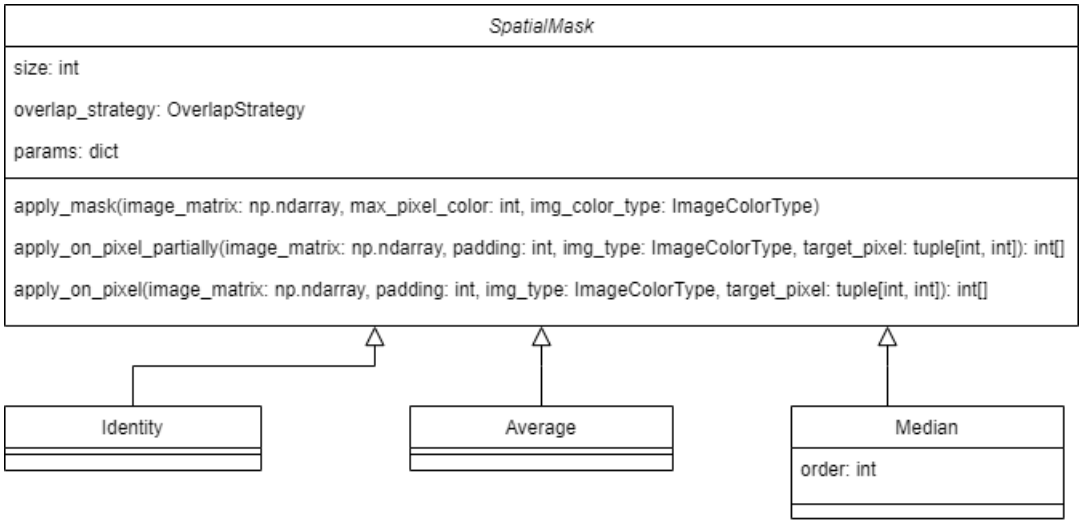
objeto do tipo *Image* nunca é alterado *in-loco*, ou seja, sempre que uma transformação T é aplicada sobre uma imagem f, o resultado é sempre uma referência à uma nova imagem, g, tal que $g = T(f)$ e f permanece inalterada. Esse comportamento facilita a exibição e comparação entre f e g sem que o utilizador explicitamente crie cópias de f.

Por fim, cabe ressaltar que operações pontuais podem ser aplicadas diretamente sobre uma *Image*, como o negativo e a limiarização.

3.2.4 Máscaras de Convolução

Máscaras de convolução podem ser de diversos tipos, possuir diversos tamanhos e formas de implementação (GONZALEZ; WOODS, 2006). A ImageX utiliza o conceito de polimorfismo e herança oriundos da Programação orientada à objetos para incorporar todos esses aspectos. Assim, toda máscara na imgX é capaz de se comportar de forma diferente, através da sobrecarregamento das funções do tipo base *SpatialMask*, conforme diagrama da figura 3.

Figura 3 – Diagrama de classes para máscaras de convolução.



Dessa forma, apesar de cada máscara poder ser implementada de forma única, para o usuário da biblioteca isso passa desapercebido, visto que a única função que o mesmo conhece é a *apply_mask(SpatialMask)* da classe *Image*, conforme figura 2.

Parte III

Apresentação do Problema e Solução

4 Apresentação e modelagem do problema

Este capítulo apresenta a forma objetiva o escopo do problema solucionado neste relatório pela biblioteca ImageX.

4.1 O problema a ser resolvido

Existem diversas situações em que uma imagem sofre alguma ação ruidosa. Em algumas destas situações pode-se melhorar a imagem através de filtros. Por exemplo, o ruído conhecido como sal e pimenta, que é causado por ruídos nas transmissões de dados. Portanto, um dos objetivos da biblioteca ImageX é solucionar diversos problemas relacionados a imagens e entre eles os ruídos sal e pimenta e, para isso, existem os filtros não-lineares que alteram a imagem sem diminuir sua resolução e uma das suas principais aplicações é justamente a remoção de ruídos. Seguindo a especificação do trabalho, o objetivo é criar um programa que recebe como entrada um arquivo que contém uma imagem binária PBM ASCII(PGM tipo P1). Essa imagem contém um texto, todo escrito na fonte arial e com o mesmo tamanho, com uma ou mais colunas. Como saída, o programa retornará uma imagem PBM contendo o texto da entrada, porém com o ruído sal e pimenta removido e com cada palavra circunscrita com retângulo(etapa 2).

4.1.1 Etapa 1

Na etapa 1, o objetivo é remover da imagem o ruído sal e pimenta e salva-la em um novo arquivo PBM. Além disso, o nome do arquivo deve seguir o padrão exposto na figura 4

Figura 4 – Padrão de nome do arquivo.

```
grupo_I_imagem_X_linhas_Y_palavras_Z.pbm
```

4.1.2 Etapa 2

Na segunda etapa, o objetivo é a contagem das linhas e colunas, circunscrever as palavras com retângulo e possíveis recursos extras.

5 Removendo ruídos da imagem

5.1 Estratégia utilizada

Para realizar a remoção dos ruídos sal e pimenta foi utilizada uma máscara de suavização, mais especificamente Máscara da Mediana com tamanho 3, ordem(percentil) 70 e estratégia de sobreposição corte.

O tamanho de máscara 3 se mostrou suficiente em testes empíricos, sendo capaz de solucionar o problema sem inserir distorções na imagem.

O percentil 70 foi escolhido, também empiricamente, após verificar que deixar sobressair alguns tons de preto podem facilitar durante as próximas etapas da resolução do problema, pois torna as palavras mais definidas e sem detalhes desnecessários.

A estratégia de sobreposição escolhida foi a de corte, pois, como as bordas mais externas da imagem não carregam informação relevante sobre o problema, podem ser descartadas.

5.2 Gerando Imagem de Testes

As imagens para o testes foram usadas utilizando uma ferramenta de geração de texto genérico. Inicialmente foram geradas três imagens diferentes, com quantidade de palavras variadas. Além disso, foi utilizado o Gimpy para inclusão do ruído sal e pimenta.

5.3 Implementação

A máscara da mediana consiste numa convolução onde, para cada pixel central, é calculada a mediana entre ele e o restante da sua vizinhança e o valor é retornado como o novo valor para o pixel central.

A complexidade desse procedimento em imagens binárias é dada pela seguinte função:

$$O(MNK^2Log(K^2)) \tag{5.1}$$

Onde, *M, N, K* se referem, respectivamente, ao número de linhas na imagem, número de colunas na imagem e o tamanho da máscara.

Entretanto, o problema traz consigo a restrição de que as imagens são compostas apenas por zeros e uns, visto que se tratam de imagens binárias. Dessa forma, realizar

o cálculo do percentil de um vetor de 0's e 1's se resume em saber se o número de 0's é maior, em uma certa porcentagem, que a quantidade de 1's. Dessa forma:

$$g(x,y) = \begin{cases} 0, & \text{se } K^2 - QuantidadeDeUns \geq \left\lfloor \dfrac{Percentil * K^2}{100} \right\rfloor \\ 1, & \text{se não} \end{cases} \tag{5.2}$$

Como realizar a contagens de 0's ou 1's leva tempo K^2, temos que a complexidade de tempo para o cálculo do percentil de uma matriz de 0's e 1's é:

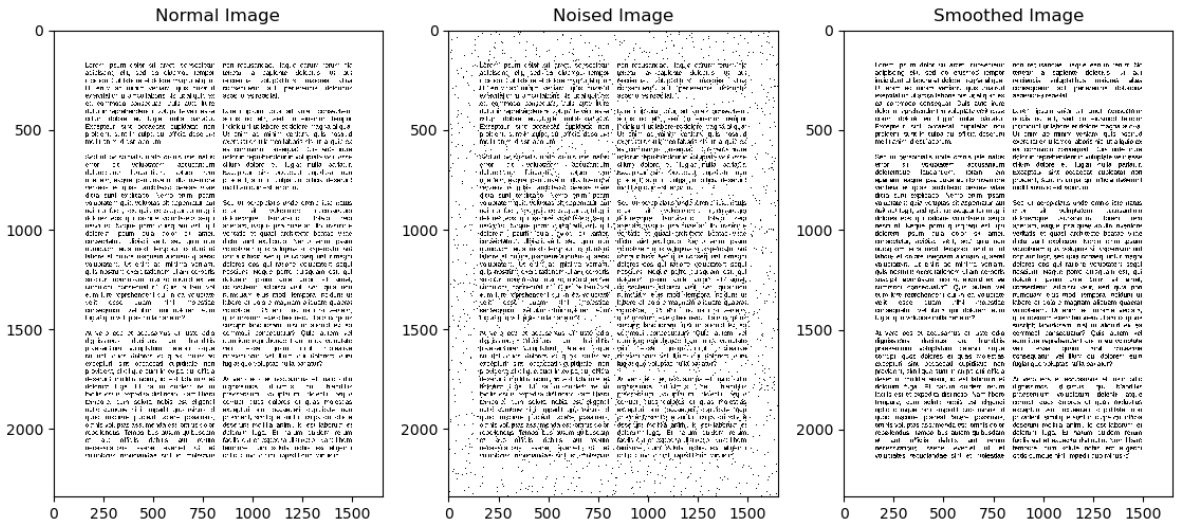
$$O(MNK^2) \tag{5.3}$$

A estratégia descrita acima foi implementada na biblioteca ImageX para imagens binárias, melhorando consideravelmente o tempo de execução da operação de convolução.

5.4 Resultados Obtidos

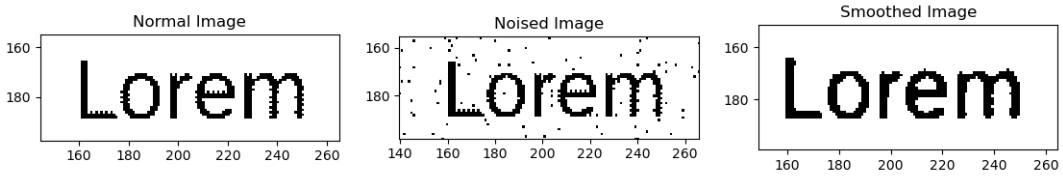
Após a aplicação da estratégia descrita nas seções acima, nota-se que o ruído sal e pimenta foi removido sem muitas deformações em outras características da imagem, como esperado ao se utilizar a máscara da Mediana com um tamanho pequeno em relação ao do total de pixels.

Figura 5 – Resultado da suavização utilizando a máscara da Mediana 70% da ImageX.



Já na figura 6, pode ser visualizado o efeito de mover a Mediana para a direita, na posição do percentil 70. A imagem resultado, suavizada, teve a intensidade dos pixels das letras realçadas, removendo alguns pixels brancos que acrescentavam detalhes na imagem original. Tais detalhes são dispensáveis para o problema e talvez possam acrescer problemas nas próximas etapas do projeto, por isso foram removidos.

Figura 6 – Detalhe do resultado da suavização da imagem



5.5 Observações e pontos de melhoria

Esta seção elenca pontos de atenção e melhoria na implementação da biblioteca e solução do problema.

5.5.1 Performance

Apesar da aplicação da melhoria do desempenho durante o cálculo da convolução da Mediana, o algoritmo ainda possui um longo tempo de execução, levando entre 25 e 67 segundos para finalizar o processo, enquanto que ferramentas como o GIMP conseguem realizar o mesmo procedimento, no mesmo conjunto de imagens, em menos de 2 segundos

Isso se deve ao fato de que nem todas as operações da ImageX são vetorizadas, inclusive as convoluções, e, uma vez que a biblioteca é escrita em uma linguagem interpretada como Python, as operações dentro de laços de repetição levam mais tempo para ser executados, pela natureza da própria linguagem (HUFFPOST, 2021).

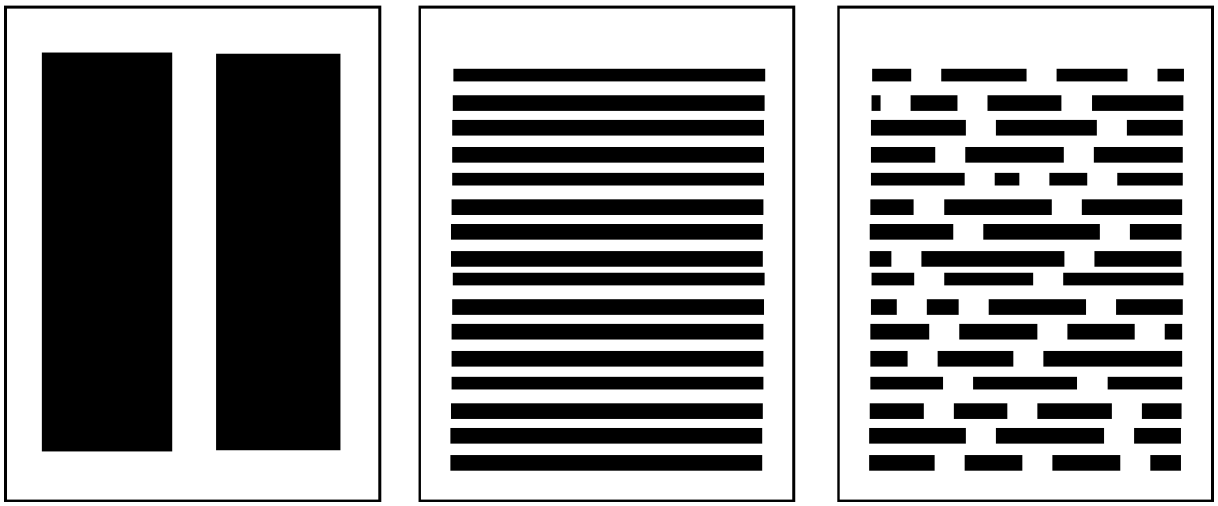
Esse gargalo pode ser corrigido utilizando-se técnicas de vetorização mais avançadas do pacote NumPy. Isso exigiria um estudo mais aprofundado do mesmo e uma refatoração na estrutura de máscaras da ImageX. Entretanto, apesar de ser possível, foge completamente do escopo da disciplina, por isso não será abordado durante a mesma para que a atenção dos participantes possa estar voltada ao conteúdo de Processamento de Imagens.

6 Contando Linhas e Colunas

6.1 Estratégia utilizada

Para realizar a contagem de linhas e colunas foram utilizados os dois dos principal métodos da morfologia em processamento de imagens a erosão e dilatação. Para isso, esse relatório apresenta o conceito de **unidade de processamento morfológica**, UPM, , que se trata de um conjunto de características dos elementos alvo do texto em processamento (colunas, linhas e palavras), que permitem realizar a contagem e classificação das mesma dentro do texto, como mostrado na Figura 6.1.

Figura 7 – Unidade de processamento morfológica para colunas, linhas e palavras (respectivamente, da esquerda para a direita)



O conjunto de características apresentado acima permite que as colunas, linhas e palavras possam ser contabilizadas e localizadas dentro do texto, enquanto que ao mesmo tempo descarta informações semânticas sobre o significado de cada elemento, que são desnecessárias durante essa etapa do processamento, reduzindo a complexidade do problema.

Dessa forma, o processo de contagem/segmentação dos elementos alvo do texto se resume em localizar suas devidas unidades de processamento morfológicas dentro do texto. Para isso foi utiliza o simples algoritmo a seguir:

1. *Para cada elemento alvo, faça:*

    a) *Execute operações de erosão e dilatação sucessivas para formar sua UPM.*

O laço acima representa a técnica geral para gerar os UPM de cada elemento alvo. Entretanto, cada elemento é individual e são necessárias sequências diferentes, com elementos estruturantes diferentes para que seja possível localizar cada um dele corretamente. As sequencias de operações morfológicas necessárias para localizar cada UPM foi encontrada através de testes empíricos e são apresentadas nas próximas seções.

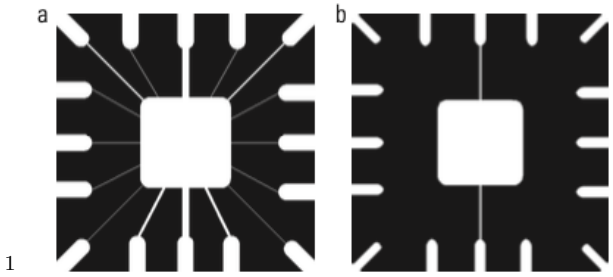
6.1.1 Erosão

A erosão segundo Gonzalez e Woods (2006) é operação representada pela formula: 6.1, ou seja é o conjunto de todos os pontos z de forma que B, transladado por z, está contido em A.

$$A \ominus B = z|(B)z \subseteq A \tag{6.1}$$

Na imagem 6.1.1, podemos ver um exemplo de uso da erosão, em que imagem fica com menos branco.

Figura 8 – Exemplo de aplicação da erosão para remover componentes da imagens



1

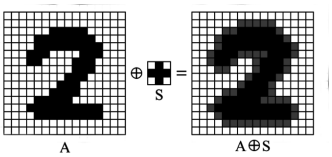
6.1.2 Dilatação

Já a dilatação, segundo Gonzalez e Woods (2006), é representando pela equação 6.2, A dilatação de A por B é, então, o conjunto de todos os deslocamentos, z, de forma que B e A se sobreponham pelo menos por um elemento.

$$A \oplus B = z|(B) \cap A \tag{6.2}$$

Na pratica a imagem parece espanar e aumentar a quantidade de pontos 1 de acordo com a mascara. na imagem 6.1.4.

Figura 9 – Exemplo de aplicação da dilatação para remover componentes da imagens



2

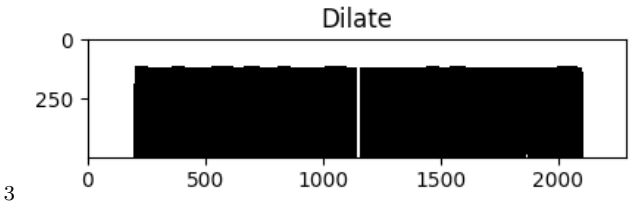
6.1.3 Contando colunas

Para contar a quantidade de colunas de um texto, foi utilizando as técnicas mencionadas acima em sequência.

A primeira técnica utilizada foi a erosão, utilizando um elemento estruturante 3x3 foi possível eliminar os ruídos das imagens. Em sequência, com o resultado da operação de erosão, foi aplicado duas dilatações uma em sequência do resultado da outra, às duas foram feitas utilizando um elemento estruturante de 79 linhas por 19 colunas, preenchidas com dígito 1.

Com isso, foi formado dois blocos pretos onde iterando entre esses bits é possível contar a quantidade de sequências de dígitos zero. É possível conferir os resultados das operações na seguinte imagem:

Figura 10 – Imagem gerada na execução do caso de teste para calculo das colunas do texto



3

Por fim, para contar as colunas foi, iterado no eixo x com os resultados das operações mencionados acima, somando mais um sempre que encontra uma sequencia de pixel pretos.

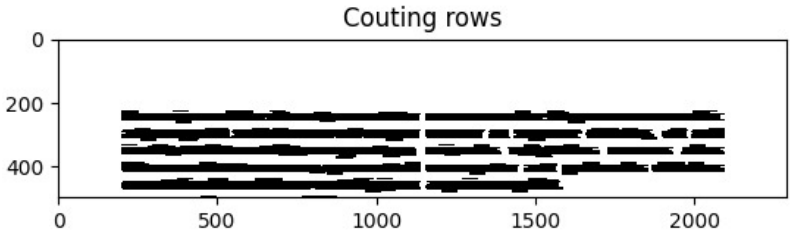
6.1.4 Contando linhas

Para contar a quantidade de linhas de um determinado texto também foi utilizado as técnicas mencionadas.

O primeiro passo foi fazer uma erosão para remover os ruídos utilizando um elemento estruturante de 3 linhas por 3 colunas preenchidas com dígitos 1.

Em sequência foi realizado uma dilatação com máscara de 3 linhas por 51 colunas, e depois uma erosão com elemento estruturante de 9 linhas por 3 colunas, com isso foi possível gerar a seguinte imagem:

Figura 11 – Imagem gerada na execução do caso de teste para calculo das linhas do texto



4

Para contar as linhas foi iterado no eixo y com os resultados das operações mencionados acima, somando mais um sempre que encontra uma sequencia de pixel pretos.

6.1.5 Detectando palavras(circunscrição com retângulo)

Para fazer a detecção de palavras foi feito uma dilatação com um elemento estruturante com 1 linha e 13 colunas e em seguida foi feito a dilatação com a transposta do elemento estruturante, com o objetivo de formar os quadrados.

Em seguida, com o resultado anterior, foi realizado uma subtração com a erosão do resultado com um elemento estruturante de 3 linhas por 3 colunas. Com o objetivo de encontrar as bordas das palavras.

Referências

BIANCHI, R. B. R. F. R. F. M. M. R. A. C. Um sistema de reconhecimento automÁtico de placas de automÓveis. *Anais do II ENIA*, p. 537–539, 1999. Citado na página 7.

ESCARPINATI, M. C. *Investigação de formatos e compressão de imagens digitais para processamento de imagens mamográficas de mamas densas*. Dissertação (Dissertação de Mestrado) — USP, São paulo, 2016. Citado na página 7.

GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing (3rd Edition)*. USA: Prentice-Hall, Inc., 2006. ISBN 013168728X. Citado 3 vezes nas páginas 7, 14 e 22.

HUFFPOST. *Computer Programming Languages: Why C Runs So Much Faster Than Python*. [S.l.], 2021. <https://www.huffpost.com/entry/computer-programming-languages-why-c-runs-so-much_b_59af8178e4b0c50640cd632e> - Acesso: 02 de Junho de 2021. Citado na página 20.

KIRSCH, R. A. et al. Experiments in processing pictorial information with a digital computer. In: *Papers and Discussions Presented at the December 9-13, 1957, Eastern Joint Computer Conference: Computers with Deadlines to Meet*. New York, NY, USA: Association for Computing Machinery, 1957. (IRE-ACM-AIEE '57 (Eastern)), p. 221–229. ISBN 9781450378628. Disponível em: <https://doi.org/10.1145/1457720.1457763>. Citado na página 7.

MAUAD, A. Através da imagem: fotografia e história–interfaces. *Tempo*, v. 1, n. 2, p. p. 73–98, 1996. Citado na página 7.

NUMPY. *What is NumPy?* [S.l.], 2021. <https://numpy.org/doc/stable/user/whatisnumpy.html> - Acesso: 02 de Junho de 2021. Citado na página 11.

SILVA, M. C. d. *Determinação do volume de madeira empilhada através de processamento de imagens digitais*. Tese (Dissertação) — Universidade Federal de Viçosa, Viçosa, 2003. Citado na página 7.