Projeto de Graduação

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

17 de junho de 2021

Processamento de imagens usando o MATLAB

Vinícius Henriques Secioso

PUC RIO

www.ele.puc-rio.br

Projeto de Graduação

DEPARTAMENTO
DE ENGENHARIA
ELÉTRICA

Processamento de imagens usando o MATLAB

Aluno: Vinícius Henriques Secioso

Orientador: Marco Antônio Grivet Mattoso Maia

Trabalho apresentado com requisito parcial à conclusão do curso de Engenharia Elétrica na Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil.

Agradecimentos

Agradeço profundamente aos meus amigos em primeiro lugar por toda ajuda, paciência, apoio, incentivo e resenha. Especialmente aos meus amigos, e colegas de curso, Breno Perlingeiro e Victoria Palhares que me abençoaram durante todo o ciclo profissional, e à Karina Santos pelo seu apoio nos meus últimos períodos. Sem minhas famílias jamais teria conseguido permanecer no curso e sou grato por isso.

Gostaria de agradecer também aos meus pais e minha irmã por terem sustentado minhas dificuldades para chegar até esta etapa.

Resumo

As tecnologias de processamento digital de imagens vêm evoluindo rapidamente nos últimos anos, principalmente devido a sua abrangência de aplicação em diferentes áreas do conhecimento humano. Existem uma gama de operações que podem ser realizadas em uma imagem visando diferentes objetivos, algumas de suas principais operações e aplicações estão explicadas neste projeto.

Neste documento são apresentados diferentes conceitos acerca da área de processamento digital de sinais. Além disso, foi utilizado um método computacional para o desenvolvimento de um aplicativo no software MATLAB em que é possível realizar diversas operações morfológicas ou de contorno em uma imagem selecionada pelo usuário. Este procedimento foi explicado neste documento e o aplicativo testado.

Palavras-chave: Processamento digital de imagens, MATLAB App designer, Operações morfológicas, Contorno, Aplicativo, Elemento estruturante, Análise de imagens

DEE DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Image processing using MATLAB

Abstract

The digital image processing technologies have been rapidly evolving in the last few years, mainly due to its scope of application in different areas of human knowledge. There is a range of operations that can be realized on an image aiming various goals, some of its main operations and applications are explained at this project.

This paper presents different concepts about the area of digital signals processing. Furthermore, a computational method was used for the development off a application on MATLAB in which is possible to realize several morphologic, or contouring, operations on a user selected image. This procedure was explained at this paper and the application was tested.

Keywords: Digital image processing, MATLAB App designer, Morphological operations, Contouring, Application, Structuring element, Image analysis

Sumário

1 Introdução **1**

2 Processamento Digital **2**

a Sinais 2

b Imagens 2

3 MATLAB *App Designer* **6**

4 Processamento morfológico **7**

a Conceitos 7

b Elemento estruturante 7

c Operações morfológicas 8

d Operações de contorno 12

5 Aplicativo desenvolvido e testagem **15**

a Aplicativo 15

b Testagem 17

6 Aplicações reais **22**

7 Conclusão e estudos futuros **24**

Referências **25**

A Código do aplicativo desenvolvido no *App Designer* **26**

Lista de Figuras

1 Diagrama de lógica das etapas do processamento dos sinais. 2
2 Imagem digital com valor da intensidade de alguns de seus pixels 3
3 Exemplo de imagem melhorada 3
4 Exemplo de imagem recuperada através do processamento 4
5 Exemplo de reconhecimento de caracteres por uma máquina 4
6 Interface do MATLAB App Designer 6
7 Exemplo de interface da janela de código do MATLAB App Designer. 6
8 Representação gráfica do conceito de encaixar (fit) e interceptar (hit) 7
9 Exemplo de elemento estruturante de dimensão 5x5 em três formatos diferentes 8
10 Imagem original, em escalas de cinza, sem nenhuma operação aplicada 9
11 Exemplo de aplicação da operação dilatação 9
12 Exemplo de aplicação da operação erosão na 'Figura 10', formato de disco 10
13 Exemplo de aplicação da operação abertura na 'Figura 10', formato de disco 10
14 Exemplo de aplicação da operação de fechar na 'Figura 10', formato de disco 11
15 Exemplo de aplicação da transformada *top-hat* na 'Figura 10', formato de disco 11
16 Exemplo de aplicação da transformada *bottom-hat* na 'Figura 10', formato de disco 12
17 Exemplo de aplicação da operação de Sobel, à esquerda é a imagem original 13
18 Exemplo de aplicação da operação de Prewitt, à esquerda é a imagem original 13
19 Interface de aplicativo para realização de operações de processamento digital de imagens, com seus comandos numerados 15
20 Testagem do aplicativo para uma operação morfológica 17
21 Foto original colorida usada nos testes 18
22 Testagem do aplicativo para uma operação morfológica em imagem colorida 18
23 Foto original colorida usada no teste de operações de contorno 19
24 Testagem do aplicativo para operação de contorno 19
25 Testagem do aplicativo para uma operação morfológica em imagem colorida negativa 20
26 Testagem do aplicativo para uma operação morfológica em imagem colorida previamente modificada 20
27 Processo computacional de identificação do sexo de ratos filhotes 22

1. Introdução

Problemas envolvendo processamento digital de sinais, no geral, requerem um trabalho experimental envolvendo simulações e testagem que pode ser extensivo. A importância do processamento digital de imagens vem crescido substancialmente, é uma forma de aplicar operações em uma imagem com o objetivo de melhorá-la ou encontrar informações úteis a partir dela.

Para que se possa fazer o estudo do processamento de imagens deve-se entender as operações morfológicas, as ferramentas de contorno e os algoritmos que proporcionam a edição das imagens.

Através da ferramenta *App Designer* encontrada no software MATLAB se pode criar, editar e executar aplicativos visando uso acadêmico e operacional. Fazendo uso desta ferramenta será mostrado o procedimento, a programação e o resultado de uma aplicação voltada para o processamento digital de imagens.

2. Processamento Digital

a. Sinais

O processamento digital de sinais (PDS) é a digitalização e manipulação matemática de sinais físicos como voz, luz, áudio, temperatura, pressão, imagens e vídeos, dessa forma pode-se aplicar funções e operações matemáticas como adição, multiplicação, divisão e diferença.

O processo se baseia na captação dos sinais citados de forma analógica, usando equipamentos como câmeras, sensores, microfones, termômetros, entre outros. A partir disso, esses sinais provenientes do "mundo físico" podem ser convertidos de sua forma analógica para a digital através de conversores. Com os sinais digitalizados é possível manipulá-los visando analisa-los, edita-los ou qualquer que seja seu objetivo, essa etapa é o chamado processamento.

Com o sinal processado pode-se utilizar o resultado digital ou reconvertê-lo para analógico, dependendo do objetivo final de quem realizou o processo. A PDS contém alguns elementos chave descritos abaixo:

- Memória do programa: armazena o programa a ser usado para o processamento dos dados digitalizados.
- Memória de dados: armazena a informação a ser processada.
- Processamento computacional: performa as operações matemáticas necessárias, acessa a memória do programa e a memória de dados.
- Entrada/Saída: Podem ser diversas funções que conectam as informações digitais ao mundo físico analógico.

Figura 1: Diagrama da lógica das etapas do processamento dos sinais.

b. Imagens

Imagem é um dos tipos de sinais que se podem ser processados, e pode ser definida, em sua forma digitalizada como uma função bidimensional $f(x,y)$, em que x e y são coordenadas espaciais e a amplitude da função f em qualquer par de coordenadas é chamada de intensidade da imagem naquele ponto (x, y).

No momento em que esses valores de *x*, *y* e amplitude são finitos pode-se considerar de que se trata de uma imagem digital. Uma imagem digital é composta, portanto, de um finito número de elementos, cada qual tem uma posição e valor específicos, o elemento mais conhecido para denotar uma imagem digital é o pixel.

Figura 2: Imagem digital com valor da intensidade de alguns de seus pixels.

As vantagens de se fazer um processamento digital se dá pela grande variedade de algoritmos que podem ser aplicados à imagem e evita problemas recorrentes da análise analógica como o aumento de ruído e distorções durante o processo. Com isso, a demanda por este tipo de conteúdo teve um aumento exponencial nos últimos anos em áreas como edição de imagens (processamento artístico), agricultura, militar, indústria e ciência médica.

As áreas fundamentais de processamento de imagens são diversas, cada uma com um método e funcionalidade diferente.

Algumas dessas áreas são:

- **Modelagem e representação**
- **Melhoria:** é o processo de alteração digital de imagens visando adquirir resultados mais apropriados para análise ou demonstração. Através do processo de melhoria é possível eliminar ruídos, iluminar a imagem, ajustar contrastes, aumentar a nitidez e outras ferramentas que ajudam a facilitar a identificação de elementos chave presentes na imagem original.

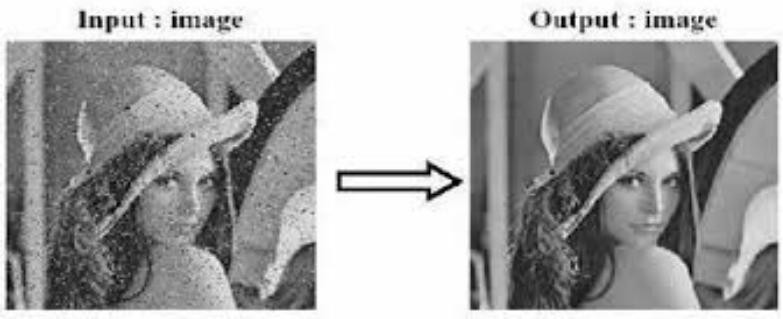


Figura 3: Exemplo de imagem melhorada.

- **Recuperação:** também chamado de restauração de imagem, é a operação em que uma figura não autêntica ou com ruído demais pode ser renovada e tornar-se mais nítida através de ferramentas de processamento, sua principal função é recuperar a perda de resolução.

Figura 4: Exemplo de imagem recuperada através do processamento.

- **Análise:** a partir de operações matemáticas ou morfológicas se pode visualizar os elementos estruturantes de uma imagem, os quais podem oferecer diversas visões que não ficavam tão evidentes originalmente, podendo, assim, contribuir em análises e conclusões da imagem modificada.

- **Reconstrução:** similar a recuperação, porém, ao invés de recuperar uma imagem ruidosa ou danificada se recorta os elementos base fundamentais da imagem original e por cima deles se constrói novos elementos que vão reconstituir a figura desejada.

- **Compressão:** é a compressão de dados aplicada a imagens digitais, visando a diminuição no custo de memória e facilitação de compartilhamento. Procedimentos podem apresentar resultados sofisticados de compressão sem perda significativa no valor e qualidade da imagem compartilhada.

- **Reconhecimento de caracteres:** é processo computacional ou eletrônico de conversão de uma imagem scaneada ou fotografada de escrita ou texto impresso em texto legível para computadores. Frequentemente é usado para registro e pesquisa de informações presentes em documentos.

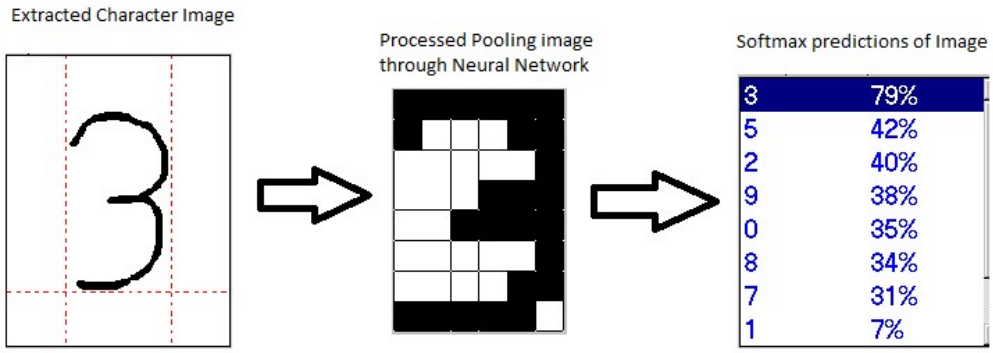


Figura 5: Exemplo de reconhecimento de caracteres por uma máquina.

Mais à frente será apresentado mais definições como as de elementos estruturantes, de operações de contorno e de operações morfológicas que foram utilizadas na elaboração de um aplicativo que tem como função auxiliar no estudo do processamento de imagens e executar essas definições em uma imagem selecionada.

3. MATLAB *App Designer*

O *App designer* é uma ferramenta presente no software MATLAB em que se torna possível a criação de aplicativos sem a necessidade de conhecimento prévio a respeito de desenvolvimento de software e aplicativos.

Através de sua interface de criação simples se pôde desenvolver um aplicativo cujo objetivo é apresentar diversas opções de operações e ferramentas de processamento digital aplicando-as a uma foto qualquer selecionada. Tornando possível realizar análises e melhorias, além de providenciar diversos exemplos para melhor entendimento das operações e das capacidades do processamento digital de imagens.

Para possibilitar o desenvolvimento de aplicações deste tipo, o software apresenta diversas opções de componentes que configuram o aplicativo e lhe garantem sua aparência. As opções são personalizáveis e possibilitam ter o design que o desenvolvedor preferir através de ações como arrastar e soltar na posição desejada e configurar a aparência da interface de usuário.

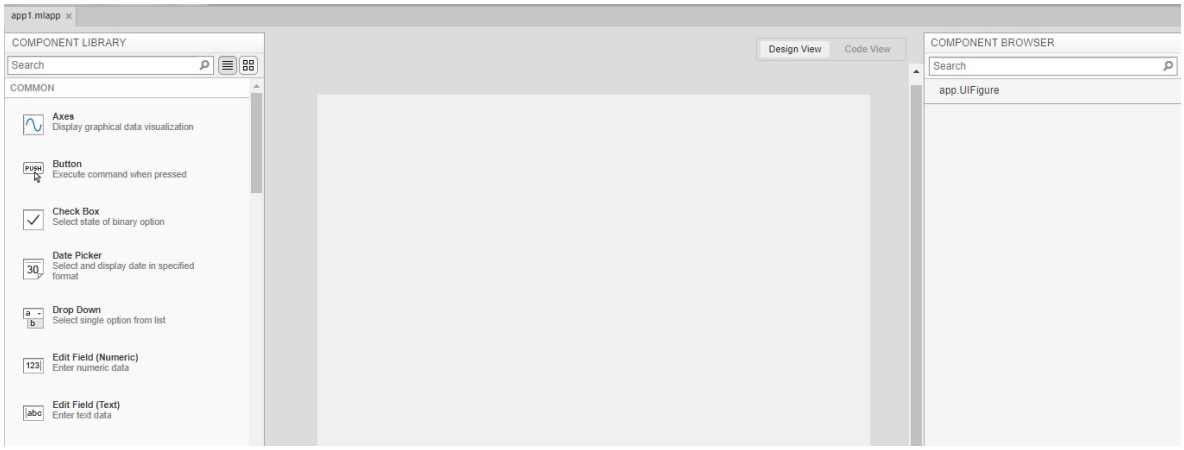


Figura 6: Interface do MATLAB App Designer

Os componentes disponíveis são os que se encontram na coluna da esquerda da imagem acima, e cada um deles é configurável e programável, na janela de código ("*code view*") é possível alterar e acrescentar funções de acordo com a biblioteca conhecida do MATLAB.

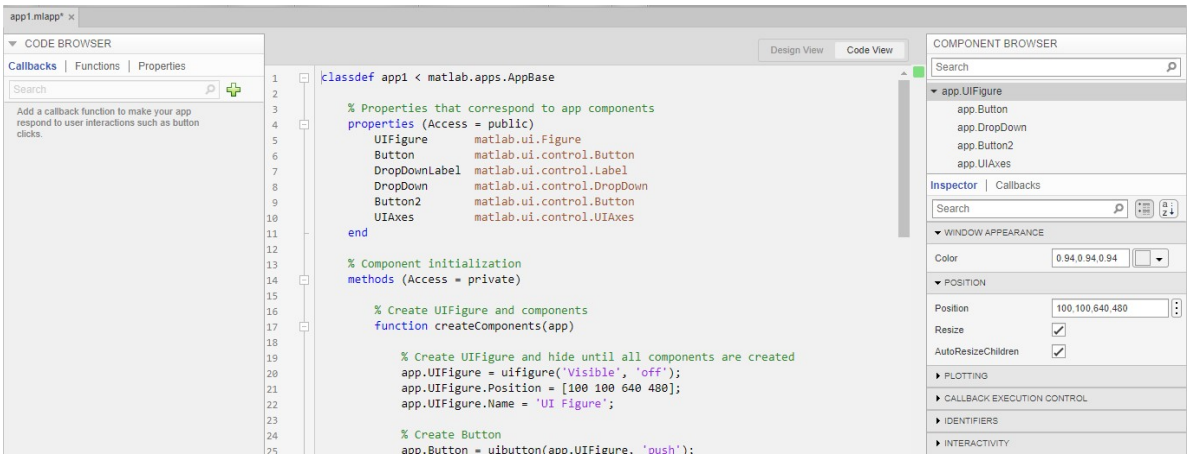


Figura 7: Exemplo de interface da janela de código do MATLAB App Designer

Fazendo uso desse software e dos conceitos de processamento digital de imagens, a serem mostrados a seguir, se tornou possível elaborar o aplicativo que será mostrado mais adiante de forma detalhada.

4. Processamento morfológico

a. Conceitos

O processamento morfológico é um conjunto algoritmos que atuam nos pixels de uma imagem de acordo com parâmetros pré-estabelecidos. Baseados nesses parâmetros, que podem ser acerca do formato ou intensidade, se pode executar diferentes operações que ajudam na forma de analisar, entender e trabalhar em uma imagem.

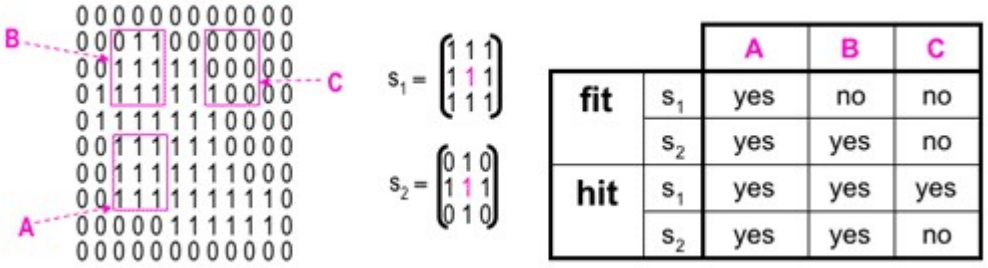
b. Elemento estruturante

Imagens digitais podem conter inúmeras imperfeições e possivelmente distorções provocadas por textura ou ruídos, o processamento morfológico tem como objetivo remover estas imperfeições através de operações na forma e estrutura da imagem.

Na matemática morfológica, um elemento estruturante é um formato usado para manipular ou interagir com uma imagem, é tipicamente utilizado para a realização das chamadas operações morfológicas, que serão especificadas adiante. Estas operações se baseiam na disposição relativa dos valores de cada pixel, não apenas em seus valores numéricos.

O elemento estruturante é posicionado em todos os locais possíveis na figura e é comparado com sua vizinhança correspondente de pixels, dessa forma as operações podem testar se o elemento "encaixa" com a vizinhança, ou testar se o elemento "atrapalha" ou intercepta a vizinhança.

Um elemento estruturante é dito como "encaixado" com a imagem se para cada pixel de valor 1, o pixel correspondente da imagem é também 1. Da mesma forma um elemento estruturante é dito que intercepta a imagem se para, ao menos, um dos seus pixels que tem valor 1 tem o pixel correspondente de valor 1 também.



		A	B	C
fit	s_1	yes	no	no
	s_2	yes	yes	no
hit	s_1	yes	yes	yes
	s_2	yes	yes	no

Figura 8: Representação gráfica do conceito de encaixar (fit) e interceptar (hit).

Pode-se entender o elemento estruturante como uma pequena imagem binária, ou seja, uma pequena matriz de pixels em que cada uma tem o valor de um ou zero. As dimensões das matrizes determinam o tamanho do elemento estruturante, enquanto o padrão de uns e zeros especifica o seu formato.

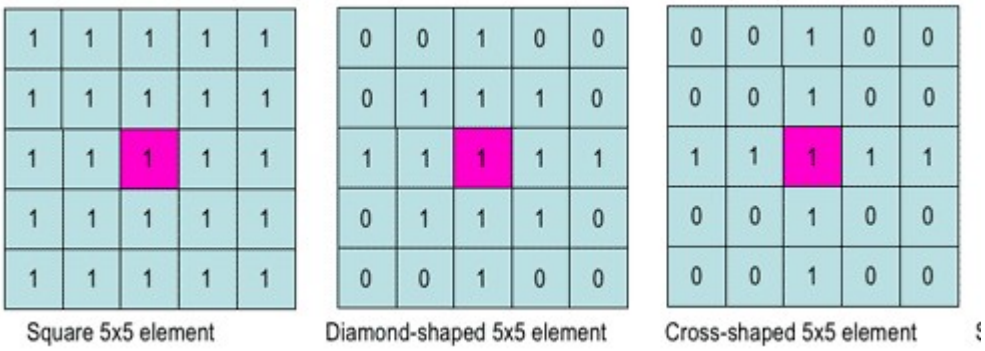


Figura 9: Exemplo de elemento estruturante de dimensão 5x5 em três formatos diferentes.

A escolha de um certo elemento estruturante para uma específica operação morfológica influencia nas informações que podem ser obtidas, tendo isto em mente, define-se as duas principais características diretamente relacionadas às operações que podem ser feitas em cada elemento.

- **Shape (Formato):** Ao escolher um elemento estruturante, há um jeito de diferenciar alguns objetos, ou partes de um objeto, presentes na imagem dos outros de acordo com seu formato ou orientação espacial. Entre os formatos possíveis de se trabalhar existe disco, diamante, quadrado, anelar, octógono, e etc.

- **Size (Tamanho):** Configurar o tamanho de um elemento estruturante é similar a configuração de uma escala de observação e permite fazer a diferenciação de objetos ou características da imagem de acordo com seu tamanho. Quanto maior o seu valor, mais distorcida ficará a imagem em comparação com a original e maior destaque se dará aos pixels de maior valor absoluto.

c. Operações Morfológicas

As operações morfológicas em processamento de sinais são um conjunto de operações não lineares relacionadas ao formato e morfologia presentes em uma imagem, e se baseiam apenas na ordenação relativa dos valores de seus pixels. São o caminho chave do processamento digital de imagens a fim de se chegar ao objetivo final desejado.

O resultado de qualquer uma dessas operações depende diretamente do formato e tamanho escolhidos para o elemento estruturante. Serão apresentadas abaixo algumas das principais operações morfológicas que podem ser aplicadas e seus exemplos:

- **Dilation (Dilatação) ($\oplus$)**

  A operação de dilatação funciona como um filtro de local máximo, ou seja, acrescenta uma camada de pixels para ambas as fronteiras interior e exterior das regiões. Sendo assim, o valor de saída de um pixel é igual ao valor máximo entre todos os pixels de sua vizinhança.

  Tal operação faz com que os objetos fiquem mais visíveis e preenche pequenos "buracos" presentes nos objetos da imagem.

Figura 10: Imagem original, em escalas de cinza, sem nenhuma operação aplicada

Figura 11: Exemplo de aplicação da operação dilatação na 'Figura 10', formato de disco.

- **Erosion (Erosão) ($\ominus$)**

  Exatamente o oposto da operação de dilatação, é usado para reduzir os formatos contidos em um objeto, sendo um filtro de local mínimo.

  Encolhe a imagem ao retirar uma camada de pixels de ambas as fronteiras interna e externa de um objeto. Sendo assim, o valor de saída de um pixel é igual ao valor mínimo entre todos os pixels em sua vizinhança.

  Tal operação faz com que sejam eliminadas "ilhas" e objetos pequenos de forma que apenas os objetos mais significativos permaneçam.

Figura 12: Exemplo de aplicação da operação erosão na 'Figura 10', formato de disco.

- **Opening (Abertura) (A o B = (A⊖B)⊕B)**

  A operação de abertura é a combinação das duas operações apresentadas acima, e é alcançada ao, primeiramente, aplicando a erosão à imagem e depois aplicando a dilatação em cima do resultado, utilizando o mesmo elemento estruturante.

  A abertura remove qualquer conexão estreita e linhas entre dois ou mais objetos. Assim sendo, ela é útil para remover objetos pequenos enquanto preserva o tamanho e o formato dos objetos mais significativos.

Figura 13: Exemplo de aplicação da operação abertura na 'Figura 10', formato de disco.

- **Closing (Fechar) (A • B = (A⊕B)⊖B)**

  Exatamente a ordem inversa da abertura. É alcançada ao, primeiramente, aplicar a operação de dilatação à imagem e depois aplicando a erosão em cima do resultado, utilizando o mesmo elemento estruturante.

Ao fechar se preenche qualquer buraco ou regiões estreitas na imagem. Assim sendo, é útil para o preenchimento desses pequenos buracos enquanto preserva o formato e tamanho dos objetos da imagem.

Figura 14: Exemplo de aplicação da operação de fechar na 'Figura 10', formato de disco.

- ***Top-hat***

  A transformada morfológica de *top-hat* se dá, primeiramente, pela execução da operação de abertura, sendo que em seguida se subtrai da imagem original esta imagem modificada após a abertura.

  Tal transformada é útil para melhorar o contraste em uma imagem de iluminação não-uniforme em escalas de cinza, além disso, pode-se isolar pequenos objetos de maior brilho presentes em uma imagem.

Figura 15: Exemplo de aplicação da transformada de *top-hat* na 'Figura 10', formato de disco.

- ***Bottom-hat***

  A transformada morfológica de *bottom-hat* se dá, primeiramente, pela execução da operação de fechar, sendo que em seguida se subtrai a imagem original da imagem modificada resultante após fechar.

  Tal transformada pode ser usada para encontrar e localizar pontos de maior intensidade em uma imagem em escalas de cinza.

Figura 16: Exemplo de aplicação do *bottom-hat* na 'Figura 10', formato de disco.

As operações morfológicas apresentadas acima são as ditas fundamentais, com exceção das transformadas de *top-hat* e *bottom-hat,* dentro do processamento digital de imagens. Porém essas que foram mencionadas não são as únicas, existem diversas outras operações que podem ser realizadas, mas que não fazem parte do escopo deste projeto.

d. Operações de contorno

Encontrar os contornos de uma imagem é um processo de identificação estrutural das linhas perimétricas dos objetos presentes, as quais ajudam na identificação de tendências e formatos dos objetos. Serão apresentados quatro tipos de operações de contorno que podem ser usadas.

- **Sobel**

  O operador de Sobel é uma das formas de detecção de limites mais usadas, é baseado na convolução da imagem com um filtro integral, pequeno e separável, nas direções horizontal e vertical, dessa forma esse operador é "leve" em termos computacionais.

  O filtro dos limites de Sobel tem a vantagem de providenciar concomitantemente a diferenciação, que garante a resposta nas fronteiras e a filtragem que reduz os ruídos do resultado.

Figura 17: Exemplo de aplicação da operação de Sobel, à esquerda está a imagem original.

- **Prewitt**

  O operador de Prewitt é similar ao de Sobel e é usado para detectar fronteiras verticais e horizontais em uma imagem. Porém, diferentemente de Sobel, este operador não coloca nenhuma ênfase nos pixels que estão mais próximos ao centro do objeto. É uma operação mais fácil de implementar computacionalmente e, consequentemente produz resultados mais ruidosos.

Figura 18: Exemplo de aplicação da operação de Prewitt, à esquerda está a imagem original.

Ao comparar as figuras 17 e 18, se percebe que o resultado das operações de contorno em uma mesma imagem e com os mesmos parâmetros são quase indistinguíveis, tendo em vista que ambas são operações que buscam encontrar as fronteiras dos objetos de uma imagem, isso faz sentido. A diferenciação destas operações se dá pela forma de se chegar ao resultado e o quanto pesa para o computador para realiza-las. O comando e parâmetros das operações de contorno são iguais:

$$[g, t] = edge(f, \ 'operação', T, \ 'dir')$$

Sendo f a imagem original, 'dir' é a direção preferencial para a detecção dos limites, podendo ser horizontal, vertical ou ambas. T é um limiar específico que determina a quantidade de detalhes que serão identificados como objetos significativos, impactando diretamento no resultado das fronteiras encontradas.

- **Roberts**

  A operação de Roberts é similar à de Sobel, porém é mais rapidamente processada (mais "leve") e, com isso, o resultado de uma imagem após realizada a operação de Roberts costuma apresentar uma maior quantidade de ruídos. É uma das formas mais

antigas de detecção de fronteiras, e a mais simples, sua limitada funcionalidade faz com que esta operação seja consideravelmente menos usada do que as outras.

- **Laplaciano de uma Gaussiana (LoG)**

  Esta operação se dá através da convolução da imagem com, como o nome já indica, um laplaciano de uma gaussiana (mostrada abaixo).

$$\nabla^2 h(r) = -\left[\frac{r^2 - \sigma^2}{\sigma^4}\right] e^{\frac{-r^2}{2\sigma^2}}$$

  Ao realizar esta convolução se provoca a melhora da imagem, reduzindo os ruídos. Utiliza os mesmos parâmetros das demais operações de contorno, porém o valor do limiar para a operação de LoG deve ser menor para alcançar o mesmo resultado.

5. Aplicativo desenvolvido e testagem

Todos os conhecimentos de processamento digital de imagens e do *App designer* apresentados, e suas definições, foram aplicados para possibilitar a criação e desenvolvimento de um aplicativo.

O aplicativo desenvolvido permite calcular diversas operações, com diferentes parâmetros, em uma imagem selecionada. É possível salvar a imagem modificada e aplicar outra operação de acordo com a preferência do usuário.

a. Aplicativo

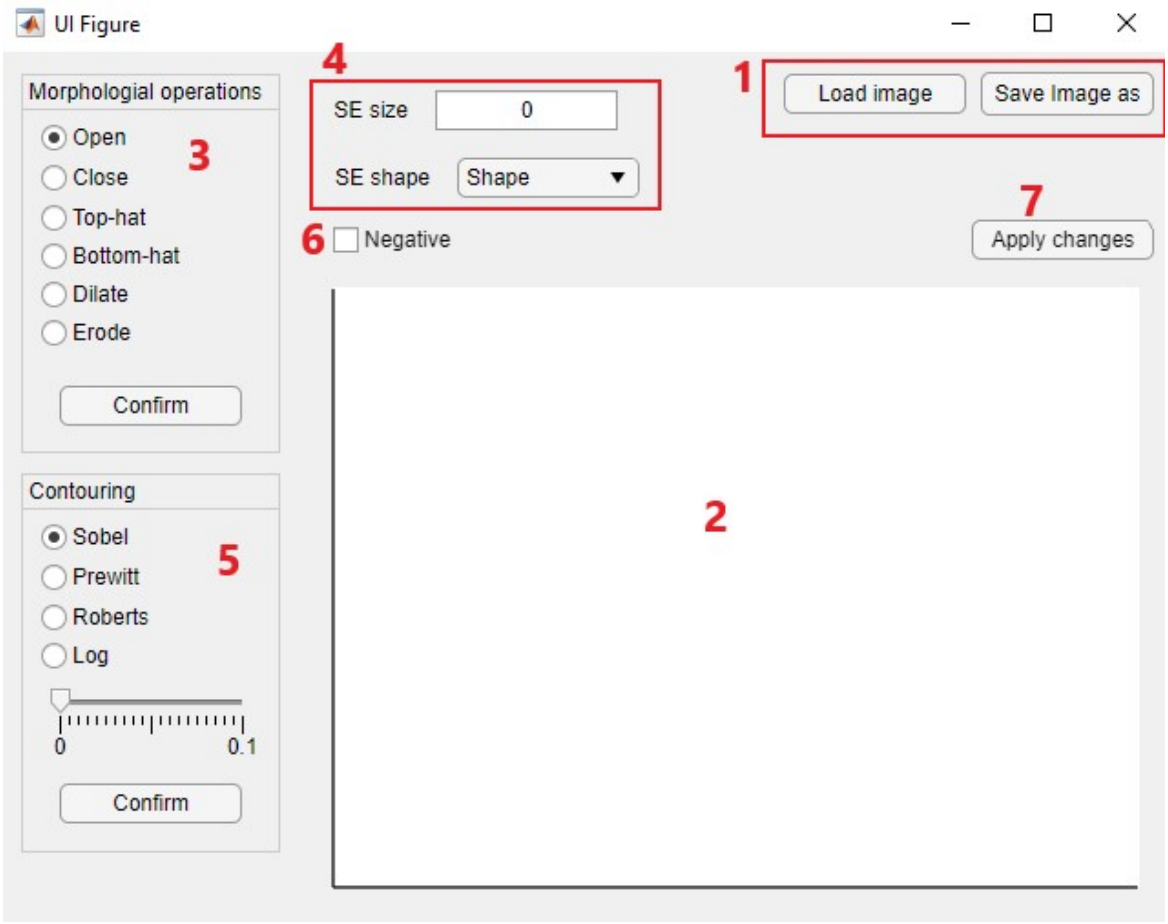


Figura 19: Interface de aplicativo para realização de operações de processamento digital de imagens, com seus comandos numerados.

A funcionalidade de cada elemento presente na interface do aplicativo é definida de acordo com a numeração de cada bloco, conforme a figura 19 acima.

1) *Load image / Save image as*

Precisamente o início e o fim de qualquer utilização do aplicativo. O botão *Load image* tem a função de carregar uma imagem selecionada a partir de qualquer pasta presente no 'explorador de arquivos' de uma máquina com o sistema operacional Windows.

O comando *Save image as* permite salvar a imagem modificada, após qualquer operação feita, na mesma pasta do em que o arquivo (.mlapp) do aplicativo se encontra.

2) Campo da imagem

Neste campo é o local onde aparecerá a imagem selecionada após o uso do botão *Load image*, e é o mesmo campo em que se visualizará as modificações feitas após qualquer nova operação realizada.

3) *Morphological operations*

É o bloco em que se encontram as opções de operações morfológicas que podem ser aplicadas na imagem selecionada. As opções foram apresentadas e definidas anteriormente.

As operações possíveis são: *open, close, erosion, dilation, top-hat* e *bottom-hat.*

Após a seleção da operação desejada e pressionar o botão *confirm,* o resultado da modificação na imagem aparecerá no campo da imagem.

4) Elemento estruturante

Neste bloco são definidos os parâmetros do elemento estruturante necessários para os cálculos das operações morfológicas.

O tamanho (*SE Size)* é definido por um número inteiro entre 0 e 100, enquanto a seleção de formato (*SE Shape*) possui três opções possíveis para o elemento estruturante, sendo eles disco (*disk*), diamante (*Diamond*) e octógono (*octagon*).

É importante salientar que para fazer a operação com o formato de octógono é necessário que o valor do tamanho (*SE Size*) seja um número múltiplo de três.

5) *Contouring*

É o bloco em que se encontram as opções de operações de contorno aplicáveis, estas alternativas foram apresentadas e definidas anteriormente.

Os métodos de contorno possíveis são: Sobel, Prewitt, Roberts e LoG (Laplaciano de uma gaussiana).

Após a seleção da operação desejada se deve determinar o valor do limiar a ser usado no cálculo da mesma, e então, se aperta o botão *confirm* para que se possa visualizar o resultado no campo da imagem.

6) *Negative*

Este comando tem a simples função de inverter as cores da imagem, ou seja, apresentar a imagem em sua forma negativa.

Ao acionar esta opção será negativada a última imagem constante no campo da imagem, seja ela modificada ou a original. Pode-se acionar este botão novamente para o esquema de cores voltar ao original, ao fazer isto, a imagem constante no campo de imagem passará a ser a imagem original, ou a última versão salva caso tenha sido aplicado o comando *Apply changes.*

7) *Apply changes*

Este botão tem a função de salvar as alterações realizadas por quaisquer operações, tornando esta imagem modificada a nova versão da imagem original. Após usar este comando toda operação feita será calculada em cima da imagem modificada salva.

b. Testagem

Serão apresentados a seguir alguns testes realizados no aplicativo desenvolvido, abordando suas principais funcionalidades.

- Teste 1: Operação morfológica

  Neste teste foi realizada a operação de *close* (fechar) com elemento estruturante de tamanho 10 e formato de disco na imagem de duas araras, a foto original consta nas figuras 17 e 18.

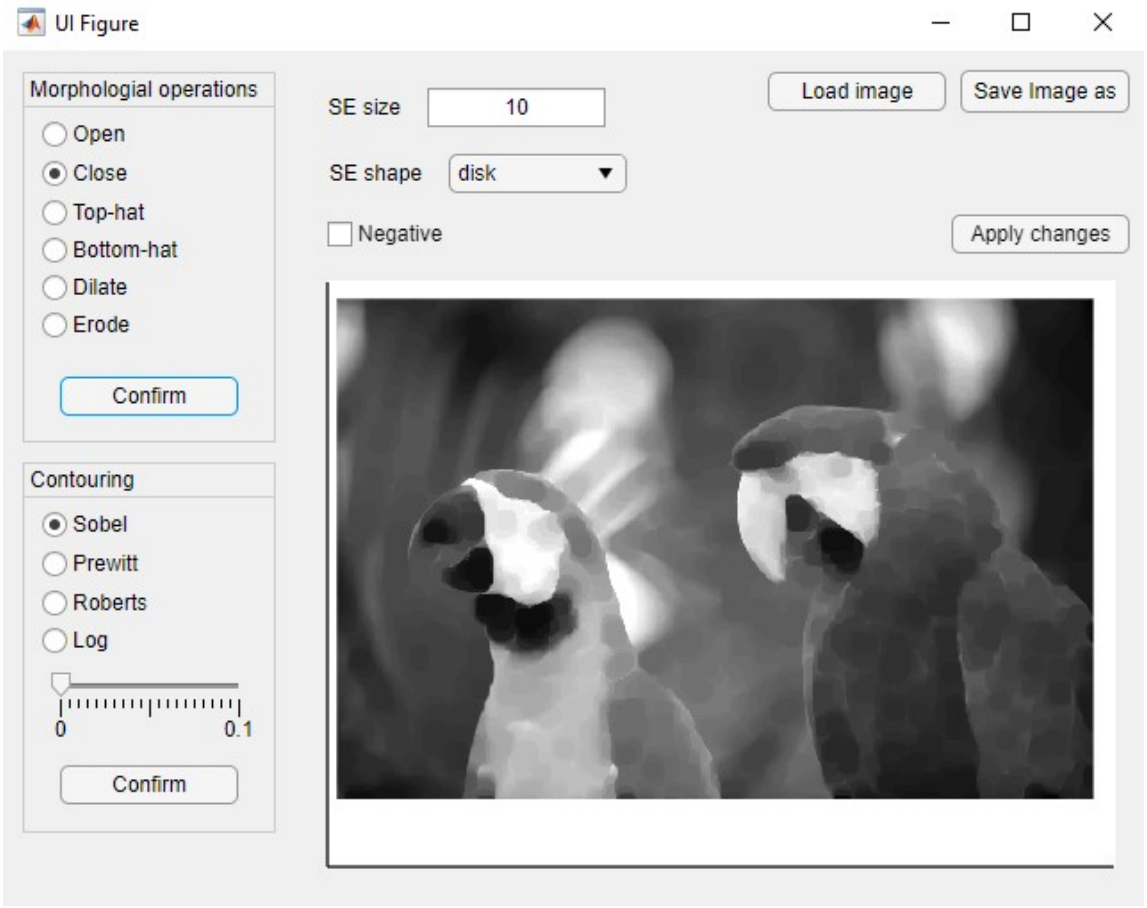


Figura 20: Testagem do aplicativo para uma operação morfológica.

- Teste 2: Imagem colorida

  Neste segundo teste foi realizada a operação de *top-hat* com elemento estruturante de tamanho 39 e formato de octógono em uma imagem colorida, cuja foto original consta abaixo:

Figura 21: Foto original colorida usada nos testes

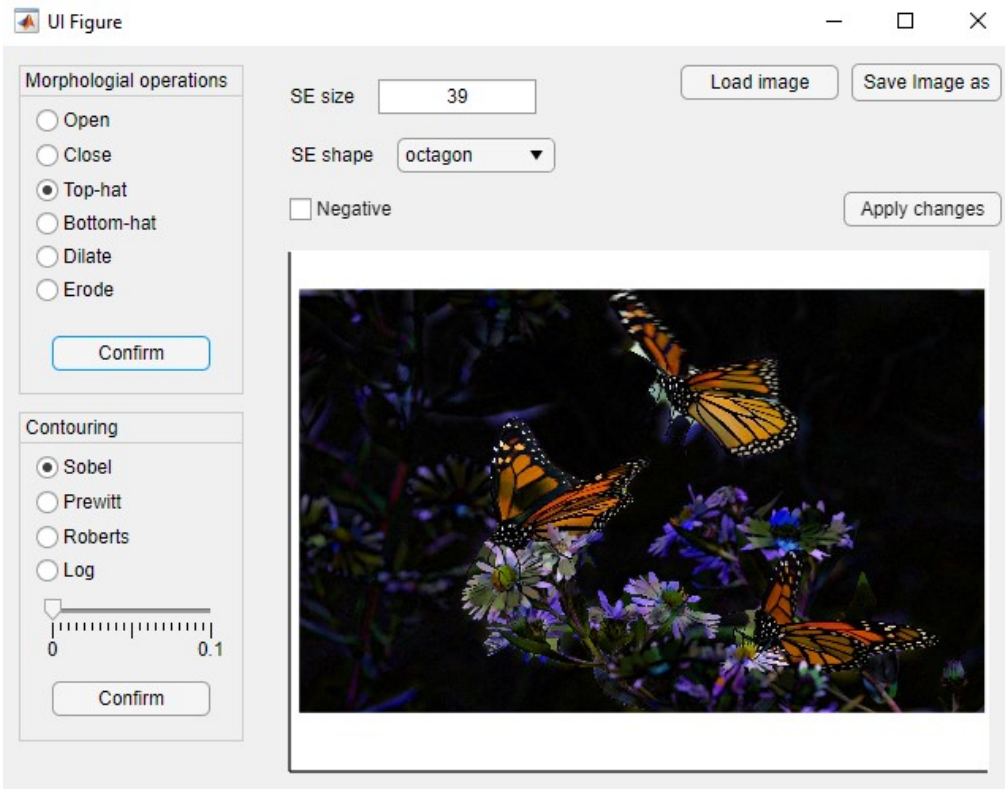


Figura 22: Testagem do aplicativo para uma operação morfológica em imagem colorida.

- Teste 3: Operações de contorno

  Neste teste foi realizada uma operação de contorno pelo método de Prewitt com valor de limiar de 0,025, aproximadamente, na imagem original mostrada abaixo.

Figura 23: Foto original colorida usada no teste de operações de contorno.

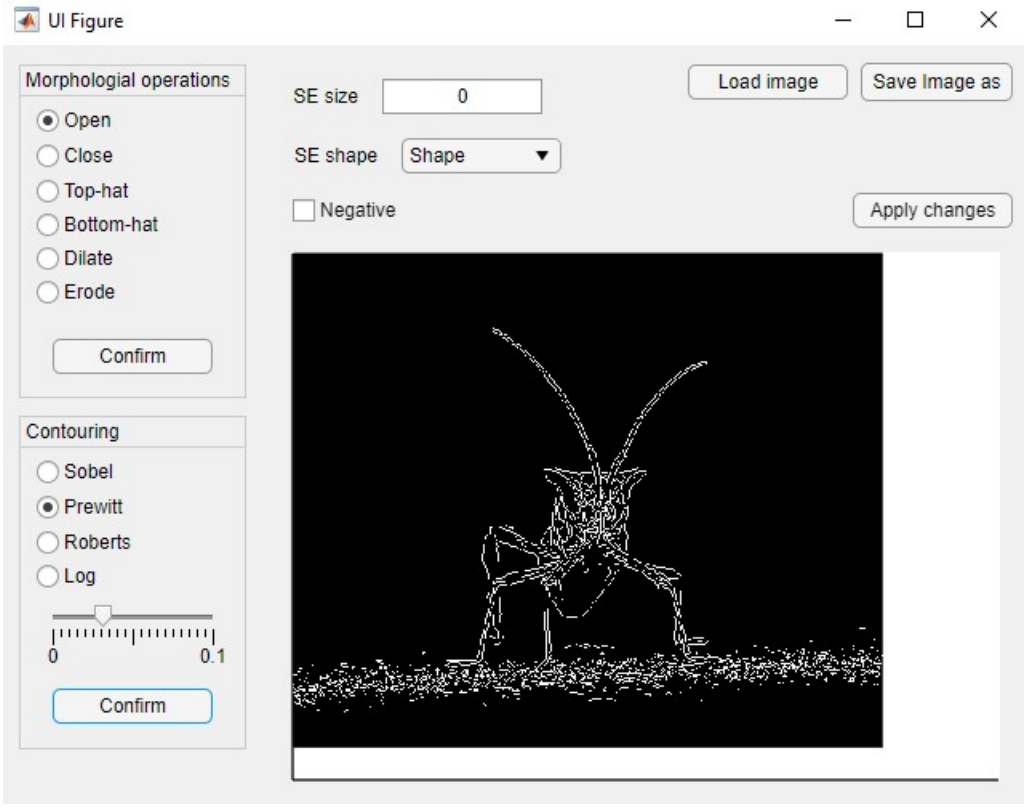


Figura 24: Testagem do aplicativo para operação de contorno.

- Teste 4: Operação em imagem negativa

  Neste teste foi realizada a operação de *dilate* (dilatar) com elemento estruturante de tamanho 11 e formato de diamante em uma imagem negativa, ou seja, uma imagem colorida com suas cores invertidas, a foto original é a figura 21.

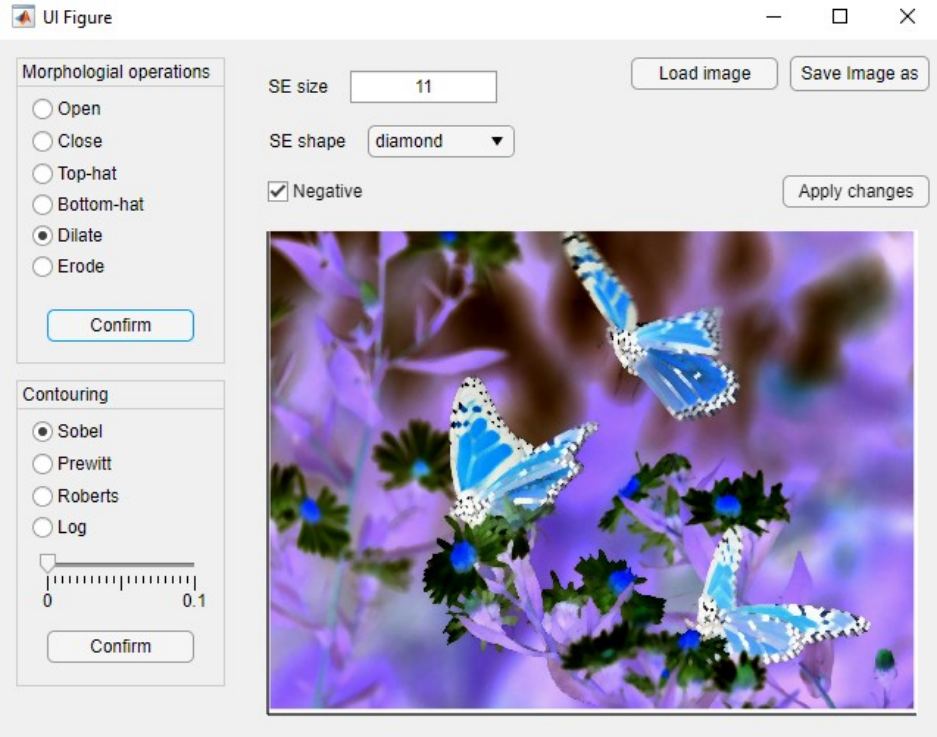


Figura 25: Testagem do aplicativo para uma operação morfológica em imagem colorida negativa.

- Teste 5: Operação em uma imagem modificada

  Neste teste foi aplicada a operação de *dilate* (dilatar) com elemento estruturante de tamanho 15 e formato de disco na imagem da figura 22.

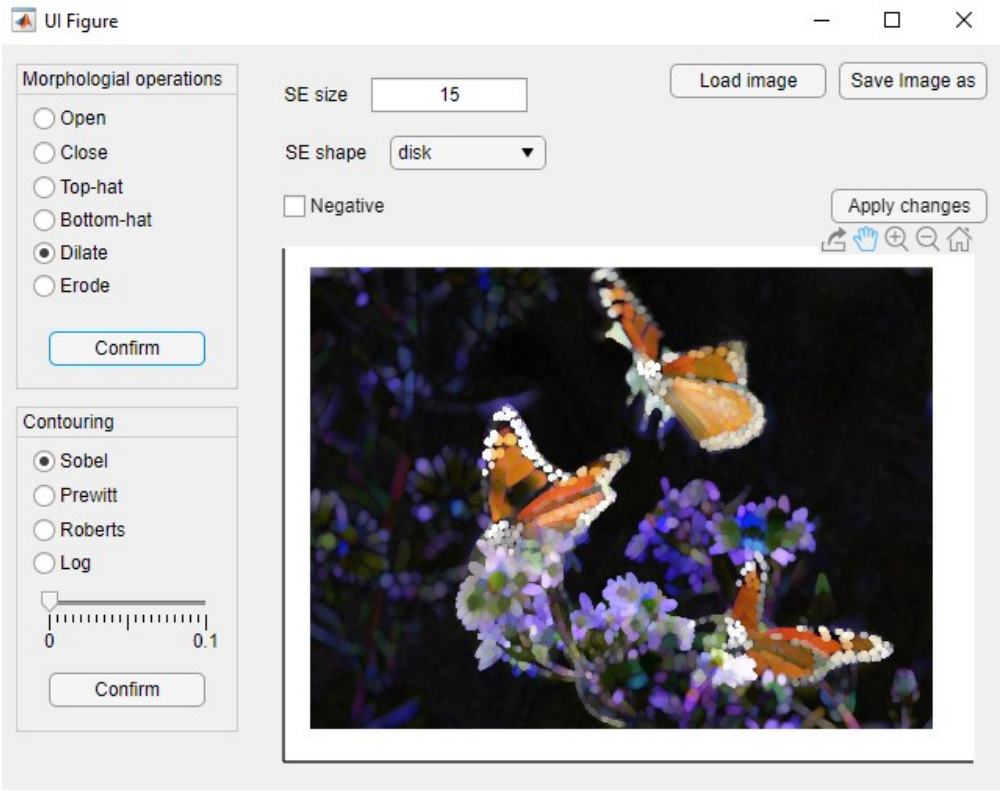


Figura 26: Testagem do aplicativo para uma operação morfológica em imagem colorida previamente modificada.

Este teste foi possível devido à função do comando *apply changes* (aplicar mudanças) em que as modificações feitas no teste 2 foram salvas, permitindo que se fizesse uma operação em cima de outra.

6. Aplicações reais

O processamento digital de sinais vem se tornado uma tecnologia desejada devido à diversidade dos seus usos, sendo uma tecnologia com o crescimento acelerado atualmente. É usada em diversas áreas como educação, entretenimento, medicina e produção.

Alguns de seus principais usos são:

→ Visão computacional: O foco dos processos é fazer as imagens serem entendidas por um computador, de forma a adquirir informações a partir delas. Este processo de aprendizado pode ser usado tanto para o controle de processos de um robô industrial e veículos automatizados quanto para contagem de público em um vídeo ou fotografia.

→ Detecção facial e verificação biométrica: O processo de identificação de uma pessoa ou uma digital através de um sistema de reconhecimento de padrões, ganhando relevância na área de segurança pública. Similar a estes sistemas, se pode usar o processamento digital como forma de detecção de íris dos olhos.

→ Análise de imagem biomédica: Crescente importância do processamento digital de imagens em diagnósticos, o objetivo é melhorar as imagens biomédicas e permitir melhores análises de exames tradicionalmente analógicos como radiografias ou endoscopias, que hoje contam com sensores digitais.

Um exemplo de aplicação real é apresentado abaixo. Neste estudo da área das ciências biológicas, o objetivo é a identificação do sexo dos filhotes de rato através de manipulações visuais em imagens desses filhotes.

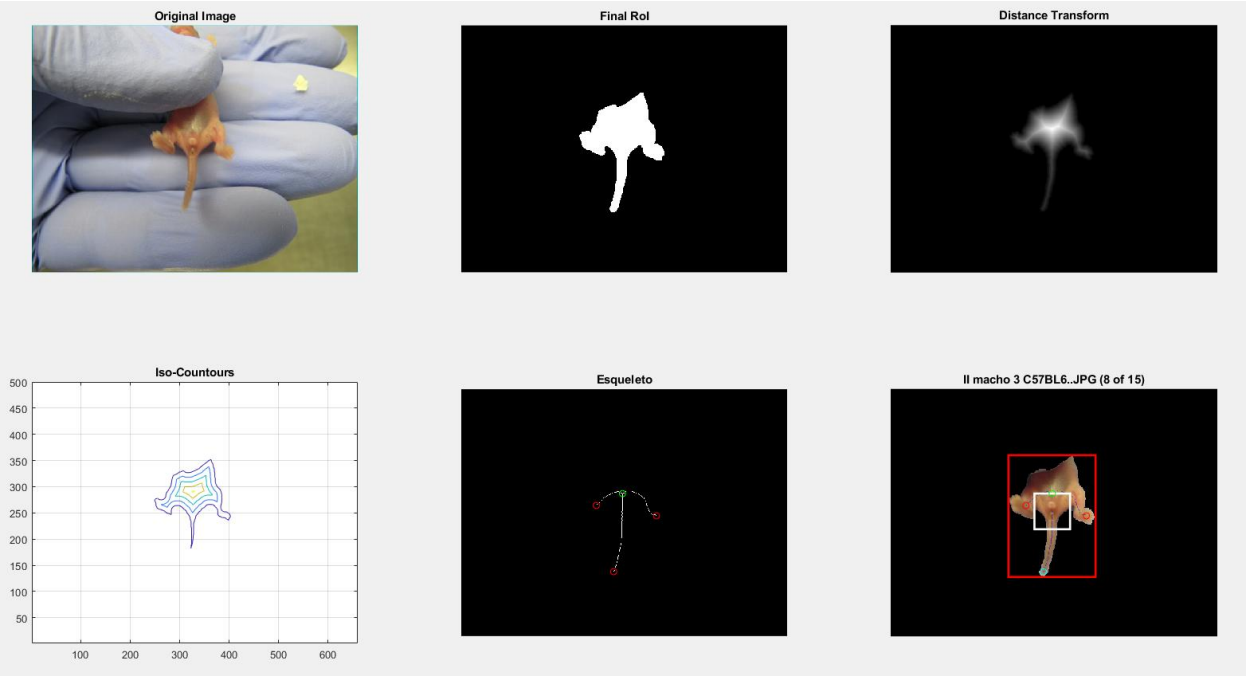


Figura 27: Processo computacional de identificação do sexo de ratos filhotes.

Para fazer esta identificação, primeiramente, foi obtida a Roi, que aparece na segunda imagem, através de um filtro cromático centrado no tom de vermelho. A seguir são aplicadas um conjunto de operações morfológicas para a suavização do seu contorno, e posteriormente, para a geração de seu esqueleto. A resposta final é encontrada a partir da distância medida entre a genitália do roedor identificada com o seu rabo.

Os usos do processamento digital de imagens se estendem por inúmeras áreas, inclusive algumas inesperadas, como na agricultura. O aplicativo desenvolvido e apresentado neste projeto tem como principal objetivo ser uma ferramenta educacional, porém, pode também ser usado na melhoria de uma imagem, na sua análise e no reconhecimento de padrões e objetos.

7. Conclusão e estudos futuros

Foram apresentados diversos conceitos, aplicações e funcionalidades a respeito da área de processamento digital de imagens. Desde a definição de conceitos gerais de processamento de sinais até o desenvolvimento de um aplicativo multi-funcional para a realização de diferentes operações em imagens.

Tendo em vista esta quantidade de informações, é interessante a possibilidade de haver outros projetos, aplicativos ou estudos complementares a este. Assim, será possível abordar novos conceitos de forma a ampliar o conhecimento geral acerca do tema. Por exemplo pode ser estudado mais a fundo a filtragem de sinais no domínio da frequência ou a segmentação de imagens.

Portato, é nítido que se trata de uma tecnologia extensa, a qual permeia diversas áreas do conhecimento humano. Sendo assim, é de interesse geral o aprofundamento desses conhecimentos e, é claro, o avanço das tecnologias de processamento digital de imagens.

Referências

[1] Rafael C. Gonzalez, Richard E. Woods and Steven L. Eddins, "*Digital Image Processing using MATLAB*", *Second Edition,* pp. 337-393

[2] DEWANGAN, S. K. Importance and Applications of Digital Image Processing. International Journal of Computer Science & Engineering Technology, p. 316–320, 2016.

[3] MAHESH INGALE, P. The importance of Digital Image Processing and its applications. International Journal of Scientific Research in Computer Science and Engineering, p. 31–32, 2018.

[4] PADMAPPRIYA, S.; SUMALATHA, K. Digital Image Processing Real Time Applications. International Journal of Engineering Science Invention, p. 46–51, 2018.

[5] TSANKASHVILI, N. Comparing Edge Detection Methods. Disponível em: <https://medium.com/@nikatsanka/comparing-edge-detection-methods-638a2919476e>. Acesso em: 3 mar. 2021.

[6] A Beginner's Guide to Digital Signal Processing (DSP). Disponível em: <https://www.analog.com/en/design-center/landing-pages/001/beginners-guide-to-dsp.html#>

[7] Morphological Image Processing. Disponível em: <https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm>

[8] Types of Morphological Operations. Disponível em: <https://se.mathworks.com/help/images/morphological-dilation-and-erosion.html>

[9] Morphological structuring element. Disponível em: <https://www.mathworks.com/help/images/ref/strel.html>

[10] Sobel Edge Detector. Disponível em: <https://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm>

[11] How Image processing is changing your world? Disponível em: <https://nu-pie.com/how-image-processing-is-changing-your-world/>

A Código do aplicativo desenvolvido no *App designer*

```
% Properties that correspond to app components
    properties (Access = public)
        UIFigure                matlab.ui.Figure
        UIAxes                  matlab.ui.control.UIAxes
        LoadimageButton         matlab.ui.control.Button
        MorphologialoperationsButtonGroup  matlab.ui.container.ButtonGroup
        OpenButton              matlab.ui.control.RadioButton
        CloseButton             matlab.ui.control.RadioButton
        TophatButton            matlab.ui.control.RadioButton
        BottomhatButton         matlab.ui.control.RadioButton
        DilateButton            matlab.ui.control.RadioButton
        ErodeButton             matlab.ui.control.RadioButton
        ConfirmButton           matlab.ui.control.Button
        SEshapeDropDownLabel    matlab.ui.control.Label
        SEshapeDropDown         matlab.ui.control.DropDown
        NegativeCheckBox        matlab.ui.control.CheckBox
        SEsizeEditFieldLabel    matlab.ui.control.Label
        SEsizeEditField         matlab.ui.control.NumericEditField
        ApplychangesButton      matlab.ui.control.Button
        SaveImageasButton       matlab.ui.control.Button
        ContouringButtonGroup   matlab.ui.container.ButtonGroup
        SobelButton             matlab.ui.control.RadioButton
        PrewittButton           matlab.ui.control.RadioButton
        RobertsButton           matlab.ui.control.RadioButton
        ConfirmButton_2         matlab.ui.control.Button
        LogButton               matlab.ui.control.RadioButton
        Slider                  matlab.ui.control.Slider
    end


    % Callbacks that handle component events
    methods (Access = private)


        % Code that executes after component creation
        function startupFcn(app)
            global storage
            storage.C_Method='Sobel';
        end


        % Button pushed function: LoadimageButton
        function LoadimageButtonPushed(app, event)
           global storage

           [filename,filepath] = uigetfile({'*.*;*.jpg;*.png;*.bmp;*.oct'},
'Select File to Open');
           fullname = [filepath, filename];
           storage.ImageFile = imread(fullname);
           colormap(app.UIAxes,'gray');
```

```
            imagesc(app.UIAxes, storage.ImageFile);
        end


        % Value changed function: NegativeCheckBox
        function NegativeCheckBoxValueChanged(app, event)
            global storage
            value = app.NegativeCheckBox.Value;

            switch value
                case 0
                    colormap(app.UIAxes,'gray');
                    imagesc(app.UIAxes, storage.ImageFile);
                case 1
                    neg = 255 - storage.ImageFile;
                    storage.ImageFileneg = neg;
                    imagesc(app.UIAxes,storage.ImageFileneg);
            end
        end


        % Button pushed function: ConfirmButton
        function ConfirmButtonPushed(app, event)
            global storage
            if app.NegativeCheckBox.Value==0


             if strcmp(app.MorphologialoperationsButtonGroup.SelectedObject.Text,
'Open')
                storage.ImageFile2 = imopen(storage.ImageFile,storage.SE);
             elseif
strcmp(app.MorphologialoperationsButtonGroup.SelectedObject.Text, 'Erode')
                    storage.ImageFile2 = imerode(storage.ImageFile,storage.SE);
             elseif
strcmp(app.MorphologialoperationsButtonGroup.SelectedObject.Text, 'Close')
                     storage.ImageFile2 = imclose(storage.ImageFile,storage.SE);
             elseif
strcmp(app.MorphologialoperationsButtonGroup.SelectedObject.Text, 'Dilate')
                     storage.ImageFile2 = imdilate(storage.ImageFile,storage.SE);
             elseif
strcmp(app.MorphologialoperationsButtonGroup.SelectedObject.Text, 'Top-hat')
                     storage.ImageFile2 = imtophat(storage.ImageFile,storage.SE);
             elseif
strcmp(app.MorphologialoperationsButtonGroup.SelectedObject.Text, 'Bottom-hat')
                     storage.ImageFile2 = imbothat(storage.ImageFile,storage.SE);
             end
            end
            if app.NegativeCheckBox.Value==1
             if strcmp(app.MorphologialoperationsButtonGroup.SelectedObject.Text,
'Open')
                storage.ImageFile2 = imopen(storage.ImageFileneg,storage.SE);
             elseif
strcmp(app.MorphologialoperationsButtonGroup.SelectedObject.Text, 'Erosion')
```

```
                    storage.ImageFile2 =
imerode(storage.ImageFileneg,storage.SE);
             elseif
strcmp(app.MorphologialoperationsButtonGroup.SelectedObject.Text, 'Close')
                  storage.ImageFile2 = imclose(storage.ImageFileneg,storage.SE);
             elseif
strcmp(app.MorphologialoperationsButtonGroup.SelectedObject.Text, 'Dilation')
                  storage.ImageFile2 = imdilate(storage.ImageFileneg,storage.SE);
             elseif
strcmp(app.MorphologialoperationsButtonGroup.SelectedObject.Text, 'Top-hat')
                  storage.ImageFile2 = imtophat(storage.ImageFile,storage.SE);
             elseif
strcmp(app.MorphologialoperationsButtonGroup.SelectedObject.Text, 'Bottom-hat')
                  storage.ImageFile2 = imbothat(storage.ImageFile,storage.SE);
             end
            end
            imagesc(app.UIAxes,storage.ImageFile2);

        end


        % Value changed function: SEsizeEditField
        function SEsizeEditFieldValueChanged(app, event)
            global storage
            storage.sesize = app.SEsizeEditField.Value;

        end


        % Value changed function: SEshapeDropDown
        function SEshapeDropDownValueChanged(app, event)
            global storage
            choice = app.SEshapeDropDown.Value;
            storage.SE = strel(choice,storage.sesize);
        end


        % Button pushed function: ApplychangesButton
        function ApplychangesButtonPushed(app, event)
            global storage
            if app.NegativeCheckBox.Value==1
                storage.ImageFile=storage.ImageFileneg;
            else
                storage.ImageFile=storage.ImageFile2;
            end
        end


        % Button pushed function: SaveImageasButton
        function SaveImageasButtonPushed(app, event)
            global storage
            fname='Imagem_Modificada.jpg';
            imwrite(storage.ImageFile,fname);
```

```
        end


        % Button pushed function: ConfirmButton_2
        function ConfirmButton_2Pushed(app, event)
            global storage
            B=edge(storage.ImageFile,storage.C_Method,storage.C_value);
            imagesc(app.UIAxes,B);
        end


        % Selection changed function: ContouringButtonGroup
        function ContouringButtonGroupSelectionChanged(app, event)
            global storage
            storage.C_Method = app.ContouringButtonGroup.SelectedObject.Text;

        end


        % Value changed function: Slider
        function SliderValueChanged(app, event)
            global storage
            storage.C_value = app.Slider.Value;

        end
    end


    % Component initialization
    methods (Access = private)


        % Create UIFigure and components
        function createComponents(app)


            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [100 100 640 480];
            app.UIFigure.Name = 'UI Figure';


            % Create UIAxes
            app.UIAxes = uiaxes(app.UIFigure);
            title(app.UIAxes, '')
            xlabel(app.UIAxes, '')
            ylabel(app.UIAxes, '')
            app.UIAxes.PlotBoxAspectRatio = [1.34375 1 1];
            app.UIAxes.XTick = [];
            app.UIAxes.YTick = [];
            app.UIAxes.YTickLabel = '';
            app.UIAxes.Position = [178 7 452 361];
```

```
            % Create LoadimageButton
            app.LoadimageButton = uibutton(app.UIFigure, 'push');
            app.LoadimageButton.ButtonPushedFcn = createCallbackFcn(app,
@LoadimageButtonPushed, true);
            app.LoadimageButton.Position = [430 447 100 22];
            app.LoadimageButton.Text = 'Load image';


            % Create MorphologialoperationsButtonGroup
            app.MorphologialoperationsButtonGroup = uibuttongroup(app.UIFigure);
            app.MorphologialoperationsButtonGroup.Title = 'Morphologial
operations';
            app.MorphologialoperationsButtonGroup.Position = [12 261 142 208];


            % Create OpenButton
            app.OpenButton =
uiradiobutton(app.MorphologialoperationsButtonGroup);
            app.OpenButton.Text = 'Open';
            app.OpenButton.Position = [11 162 58 22];
            app.OpenButton.Value = true;


            % Create CloseButton
            app.CloseButton =
uiradiobutton(app.MorphologialoperationsButtonGroup);
            app.CloseButton.Text = 'Close';
            app.CloseButton.Position = [11 140 65 22];


            % Create TophatButton
            app.TophatButton =
uiradiobutton(app.MorphologialoperationsButtonGroup);
            app.TophatButton.Text = 'Top-hat';
            app.TophatButton.Position = [11 118 65 22];


            % Create BottomhatButton
            app.BottomhatButton =
uiradiobutton(app.MorphologialoperationsButtonGroup);
            app.BottomhatButton.Text = 'Bottom-hat';
            app.BottomhatButton.Position = [11 97 81 22];


            % Create DilateButton
            app.DilateButton =
uiradiobutton(app.MorphologialoperationsButtonGroup);
            app.DilateButton.Text = 'Dilate';
            app.DilateButton.Position = [11 76 53 22];


            % Create ErodeButton
            app.ErodeButton =
uiradiobutton(app.MorphologialoperationsButtonGroup);
```

```
            app.ErodeButton.Text = 'Erode';
            app.ErodeButton.Position = [11 55 54 22];


            % Create ConfirmButton
            app.ConfirmButton = uibutton(app.MorphologialoperationsButtonGroup,
'push');
            app.ConfirmButton.ButtonPushedFcn = createCallbackFcn(app,
@ConfirmButtonPushed, true);
            app.ConfirmButton.Position = [21 15 100 22];
            app.ConfirmButton.Text = 'Confirm';


            % Create SEshapeDropDownLabel
            app.SEshapeDropDownLabel = uilabel(app.UIFigure);
            app.SEshapeDropDownLabel.HorizontalAlignment = 'right';
            app.SEshapeDropDownLabel.Position = [178 401 58 22];
            app.SEshapeDropDownLabel.Text = 'SE shape';


            % Create SEshapeDropDown
            app.SEshapeDropDown = uidropdown(app.UIFigure);
            app.SEshapeDropDown.Items = {'disk', 'octagon', 'diamond', 'Shape'};
            app.SEshapeDropDown.ValueChangedFcn = createCallbackFcn(app,
@SEshapeDropDownValueChanged, true);
            app.SEshapeDropDown.Position = [251 401 100 22];
            app.SEshapeDropDown.Value = 'Shape';


            % Create NegativeCheckBox
            app.NegativeCheckBox = uicheckbox(app.UIFigure);
            app.NegativeCheckBox.ValueChangedFcn = createCallbackFcn(app,
@NegativeCheckBoxValueChanged, true);
            app.NegativeCheckBox.Text = 'Negative';
            app.NegativeCheckBox.Position = [183 367 69 22];


            % Create SEsizeEditFieldLabel
            app.SEsizeEditFieldLabel = uilabel(app.UIFigure);
            app.SEsizeEditFieldLabel.HorizontalAlignment = 'right';
            app.SEsizeEditFieldLabel.Position = [178 438 46 22];
            app.SEsizeEditFieldLabel.Text = 'SE size';


            % Create SEsizeEditField
            app.SEsizeEditField = uieditfield(app.UIFigure, 'numeric');
            app.SEsizeEditField.Limits = [0 100];
            app.SEsizeEditField.ValueChangedFcn = createCallbackFcn(app,
@SEsizeEditFieldValueChanged, true);
            app.SEsizeEditField.HorizontalAlignment = 'center';
            app.SEsizeEditField.Position = [239 438 100 22];


            % Create ApplychangesButton
```

```
            app.ApplychangesButton = uibutton(app.UIFigure, 'push');
            app.ApplychangesButton.ButtonPushedFcn = createCallbackFcn(app,
@ApplychangesButtonPushed, true);
            app.ApplychangesButton.Position = [533 367 100 22];
            app.ApplychangesButton.Text = 'Apply changes';


            % Create SaveImageasButton
            app.SaveImageasButton = uibutton(app.UIFigure, 'push');
            app.SaveImageasButton.ButtonPushedFcn = createCallbackFcn(app,
@SaveImageasButtonPushed, true);
            app.SaveImageasButton.Position = [538 446 95 24];
            app.SaveImageasButton.Text = 'Save Image as';


            % Create ContouringButtonGroup
            app.ContouringButtonGroup = uibuttongroup(app.UIFigure);
            app.ContouringButtonGroup.SelectionChangedFcn =
createCallbackFcn(app, @ContouringButtonGroupSelectionChanged, true);
            app.ContouringButtonGroup.Title = 'Contouring';
            app.ContouringButtonGroup.Position = [12 42 142 208];


            % Create SobelButton
            app.SobelButton = uiradiobutton(app.ContouringButtonGroup);
            app.SobelButton.Text = 'Sobel';
            app.SobelButton.Position = [11 162 58 22];
            app.SobelButton.Value = true;


            % Create PrewittButton
            app.PrewittButton = uiradiobutton(app.ContouringButtonGroup);
            app.PrewittButton.Text = 'Prewitt';
            app.PrewittButton.Position = [11 140 65 22];


            % Create RobertsButton
            app.RobertsButton = uiradiobutton(app.ContouringButtonGroup);
            app.RobertsButton.Text = 'Roberts';
            app.RobertsButton.Position = [11 118 65 22];


            % Create ConfirmButton_2
            app.ConfirmButton_2 = uibutton(app.ContouringButtonGroup, 'push');
            app.ConfirmButton_2.ButtonPushedFcn = createCallbackFcn(app,
@ConfirmButton_2Pushed, true);
            app.ConfirmButton_2.Position = [21 16 100 22];
            app.ConfirmButton_2.Text = 'Confirm';


            % Create LogButton
            app.LogButton = uiradiobutton(app.ContouringButtonGroup);
            app.LogButton.Text = 'Log';
            app.LogButton.Position = [11 97 65 22];
```

```
            % Create Slider
            app.Slider = uislider(app.ContouringButtonGroup);
            app.Slider.Limits = [0 0.1];
            app.Slider.MajorTicks = [0 0.05 0.1];
            app.Slider.MajorTickLabels = {'0', ' ', '0.1'};
            app.Slider.ValueChangedFcn = createCallbackFcn(app,
@SliderValueChanged, true);
            app.Slider.Position = [21 81 100 3];


            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end


    % App creation and deletion
    methods (Access = public)


        % Construct app
        function app = APP_ImageProcessing_ViniciusSecioso


            % Create UIFigure and components
            createComponents(app)


            % Register the app with App Designer
            registerApp(app, app.UIFigure)


            % Execute the startup function
            runStartupFcn(app, @startupFcn)


            if nargout == 0
                clear app
            end
        end


        % Code that executes before app deletion
        function delete(app)


            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```