Algoritmos de binarização, derivação e filtragem

Aplicação no processamento digital de imagem

Autores

Ana Magalhães up202007440 MIB1F
Ana Pinto up202008182 MIB1F
Inês Calmeiro up202007385 MIB1F
José Santos up202007566 MIB1F

Data de entrega: 29-01-2021
Unidade Curricular: Introdução à Programação Científica
Mestrado Integrado em Bioengenharia

Conteúdo

Introdução

O processamento digital de imagens é uma das áreas da programação cuja interdisciplinaridade é mais evidente. Uma imagem digital é a representação bidimensional de um sinal discreto, e é representada no MATLAB através de um *array* ou matriz cujos elementos são os pixels, sendo que um pixel se caracteriza por 8 bits e pode assumir 256 valores, conforme a luminosidade. Por esse motivo, os termos "imagem" e "matriz" serão usados de forma indiferenciada ao longo deste relatório.

O objetivo primário consiste na segmentação semântica de imagens, ou seja, em criar um programa que reconheça numa dada imagem a existência de células e seja capaz de retornar informação relevante acerca das mesmas, com base em operações de binarização, derivação e filtragem. Para testar a eficiência do código, foram fornecidas três análises microscópicas de culturas de células.

Após a leitura de uma imagem, procede-se à conversão da mesma para um formato monocromático usando a função **rgb2gray.** Deixa-se, assim, de ter um array multidimensional (512x512x3 uint8) para ter um unidimensional.

A função **my_norm** visa a normalização das imagens de modo a tomarem valores conhecidos entre 0 e 1. Previamente, no entanto, teve de ser convertida para o formato double para que pudessem assumir valores decimais (caso contrário estar-se-ia, involuntariamente, a binarizar a imagem). Assim, a imagem resultante é guardada numa variável 512x512 double.

Foram implementadas diversas funções que permitiram a interpretação das imagens: com a função **my_hist** gera-se o histograma da imagem monocromática possibilitando a interação do utilizador na definição de um intervalo de intensidades para as diferentes partes das células e do fundo. O histograma representa a frequência relativa de ocorrência dos diferentes níveis de cinza na imagem.

Codificou-se ainda uma função **my_bin** que depende do intervalo definido pelo utilizador e da qual resulta numa imagem binarizada, mantendo-se as intensidades nas áreas de interesse enquanto que os restantes pixels serão representados a branco. É possível estender a função **my_bin** à obtenção de uma imagem segmentada. Posteriormente foram implementadas duas funções, **my_median** e **my_mean,** que são responsáveis pela aplicação de um filtro de mediana e de média, respetivamente. A imagem final após o teste do filtro de mediana apresenta uma

atenuação dos ruídos presentes face à imagem pré-processada, enquanto a média realiza apenas uma suavização uniforme.

A função **my_deriv** calcula a derivada de 1ª ordem ao longo das linhas e colunas da imagem. Criou-se também a função **my_text** que, após a inserção de texto pelo utilizador, exibe nas coordenadas apontadas pelo cursor o referido texto. Com as mesmas bases foi formulada a função **get_info** que apresenta na imagem original (a cor) os valores de máximo e mínimo dados pela **my_deriv** e, numa caixa de texto retangular, informação relacionada com o pixel selecionado.

Foram ainda usadas várias funções auxiliares para organizar o fluxo de informação ao longo do código, na tentativa de minimizar o risco de repetições. Além disso, englobando todas as funções acima descritas, foi criado um sistema de processamento de imagem controlado pelo utilizador, que pode selecionar diferentes operações que pretenda ver executadas e guardar o seu resultado.

Algoritmo de avaliação do histograma

A função **my_hist** permite a elaboração de um histograma da imagem monocromática, que fornece uma indicação da qualidade da mesma quanto ao contraste e intensidade luminosa. utilizando uma função nativa do matlab - **histogram**. O histograma foi otimizado, adequando-se o número de colunas aos 256 valores de intensidade da imagem.

Posto isto, é necessária uma avaliação do histograma produzido. Verificou-se, contudo, que algumas imagens produzem histogramas bastante ambíguos, o que dificulta a sua correta interpretação.

Por este motivo, e como forma de valorizar o código, o utilizador pode optar por receber sugestões, que o orientam indicando possíveis valores de intensidade que representam regiões diferentes. Estes valores são definidos pela função **inflection**.

Os dados do histograma podem ser representados, com o auxílio das funções **polyfit e polyval**, como um polinómio (de grau 5, que após vários testes se considerou ser a melhor aproximação), o qual será possível derivar duas vezes e, por conseguinte, descobrir as aproximações, pelos valores mínimos, aos zeros da segunda derivada, ou seja, aos pontos de inflexão do gráfico que representa a informação dada pelo histograma.

O fundamento desta função é que se pode inferir que tais pontos correspondem a variações importantes na quantidade de pixels numa gama de intensidades, logo muito provavelmente delimitam regiões diferentes na imagem. É, então, com base neles que é realizada a sugestão ao utilizador do intervalo de intensidades correspondente às células.

Após realizar testes com as imagens disponibilizadas verificou-se que esta função tem efetivamente algum potencial. Por exemplo, na imagem celula#1.png permitiu obter limites (0.12 e 0.42 na Figura 1) que ao ser automaticamente colocados na função **my_bin** produzem uma imagem que separa a célula do fundo com sucesso. Contudo, dado que este sucesso não se generalizou a todas as imagem testadas é importante reconhecer que apesar do potencial a função pode ser melhorada.

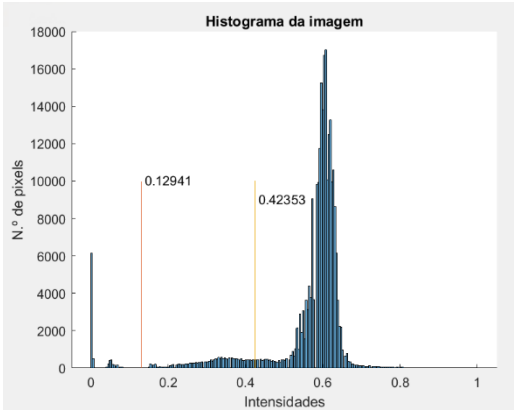


Figura 1 – Histograma com sugestões do ficheiro celula#1.png

Algoritmos de binarização da imagem

A função **my_bin** representou a introdução de um nível significativo de liberdade ao utilizador, para além de ser, talvez, o passo mais importante para o alcance do propósito principal: a segmentação semântica. Os parâmetros de entrada utilizados para a binarização resultam da análise do histograma por parte do utilizador, quer este se tenha, ou não, servido das sugestões automáticas. Este intervalo de intensidades, com uma correta interpretação, corresponde às áreas de interesse - células. Realiza-se, então, *Image thresholding* com dois limiares.

Caso o utilizador pretenda uma imagem com tons originais nas áreas do intervalo, o programa efetua aquilo que é pedido no guião: as regiões de interesse mantêm os níveis de intensidade originais, sendo que todos os outros valores irão representar o fundo da imagem, cujas intensidades são substituídas por 1-cor branca.

Por outro lado, se for selecionada a opção "Preto e Branco" no sistema de processamento, produz-se uma imagem cujos pixels do intervalo são brancos, sendo os restantes pretos, segundo os mesmos princípios.

Pode optar-se, adicionalmente, por pós-processar a imagem, fim para o qual se implementou a função **proposbin**. Esta função usa a matriz binarizada com 0's e 1's e tem como finalidade eliminar todo o ruído que seja demasiado grosseiro. Cada pixel é substituído pelo inteiro (0 ou 1) mais próximo da média dos valores da sua vizinhança, num processo que pode ser repetido várias vezes até se obter resultados satisfatórios. Os parâmetros de entrada que regulam a dimensão da vizinhança e o número de repetições podem ser inseridos pelo utilizador ou podem usar-se valores automáticos, resultantes, mais uma vez, de verificação por testes ao programa.

Embora seja possível que esta extensão da binarização cause alguma distorção da forma das células, ela facilita consideravelmente o processo de segmentação semântica, logo, na função **segmen**, efetua-se a chamada à **my_bin** ativando a opção do processamento pós-binarização.

Em **segmen**, que corresponde a uma das opções principais no sistema de processamento, o utilizador pode não só definir os limiares de intensidades (que serão úteis para a binarização) como também selecionar com o cursor todas as regiões que são do seu interesse, o que se traduz no código por um ciclo while. Os pixels correspondentes a estas áreas são armazenados como regiões diferentes com recurso à função nativa **bwlabel**.

A realização da segmentação semântica possibilita a obtenção de outras propriedades relacionadas com cada célula, que remetem para o capítulo "Algoritmo de apresentação de texto e cor". Isto será, opcional, e a escolha recairá sobre o utilizador após ter realizado segmentação.

Algoritmos de filtragem da imagem

A aplicação de filtros de vizinhança a uma imagem traduz-se, em termos práticos, na substituição da intensidade de cada pixel da imagem original (ou pré-processada, no caso), por um novo valor, dependente do próprio e das intensidades dos pixels vizinhos.

Estes algoritmos implicam, portanto, a definição de uma vizinhança para cada ponto, considerada para o efeito como uma matriz quadrada cuja ordem corresponde à dimensão do filtro. No caso de esta ser ímpar, o ponto central – onde será guardado o resultado da operação – é de determinação intuitiva. Numa vizinhança de dimensão par, tem-se que o pixel central fica imediatamente acima e à esquerda do centro geométrico. No geral, para uma vizinhança m-por-n, o pixel central é floor(([m n]+1)/2).[11]

De forma a evitar a repetição de código comum e aumentar a eficiência, criou-se uma função global **-my_filtro**. A dimensão da vizinhança do filtro é definida pelo utilizador, e é tratada de forma diferente conforme a sua paridade ou imparidade. Os valores da vizinhança de cada ponto são organizados num vetor, recorrendo à função **vetviz.**

Através de um comando switch, são invocadas várias funções auxiliares especificamente à média ou à mediana. O parâmetro que varia neste switch é introduzido pelas funções **my_mean** e **my_median** (2 ou 1, respetivamente) conforme a escolha do utilizador.

Ambos os filtros utilizados neste trabalho implicam um tratamento especial das bordas da imagem, para evitar que pontos exteriores à mesma sejam considerados como vizinhança. Para tal, após o processamento dos pixels do centro da imagem é executada a função **analiseborda**, que processa as bordas da imagem região a região. A cada uma destas regiões atribuiu-se uma variável, tal como ilustrado na Figura 2. As matrizes destas bordas e a imagem central são, posteriormente, concatenadas.

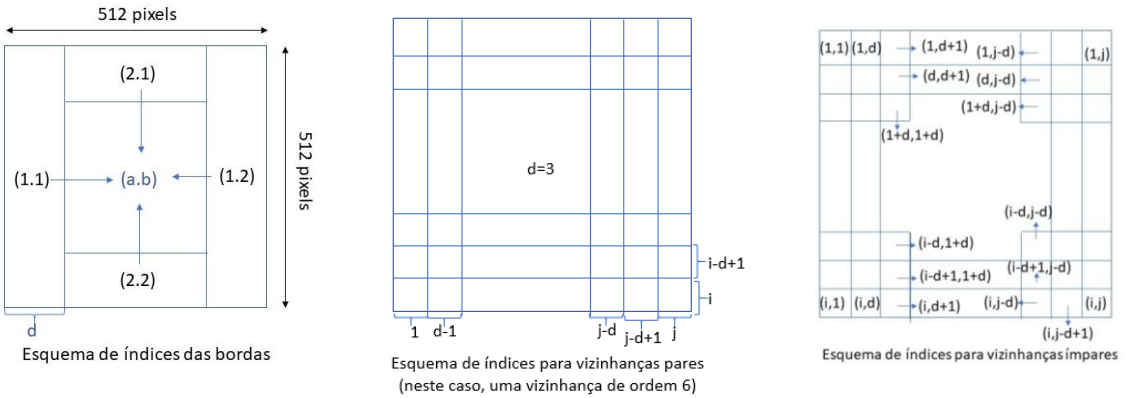


Figura 2- Esquemas de índices para a função analiseborda

Embora nas funções **my_mean** e **my_median** haja apenas a chamada à função **my_filtro**, neste relatório serão apresentadas as especificidades de cada uma das primeiras nos respetivos subcapítulos, com o propósito único de as organizar por temas.

Função my_mean

O filtro de média (não ponderada) será eficaz no sentido de suavizar a imagem, dado que os elementos da matriz resultante terão intensidades mais semelhantes às dos elementos vizinhos.

Função my_median

Uma vantagem da mediana relativamente à média prende-se com o facto de pequenos aglomerados de pixels isolados (ruído) cuja intensidade difira bastante em relação à vizinhança poderem ser removidos, ou seja, caso mais de metade dos pixels da vizinhança pertençam ao fundo, o pixel em questão ficará com intensidade semelhante.

Para a ordenação dos valores da vizinhança, usou-se o método da seleção através da função **ordenar**, por se verificar ser mais eficiente do que o método da troca, ou *bubble sort*. A mediana de cada vetor, já ordenado, é calculado recorrendo à função **medianandes** (mediana de n valores desconhecidos).

Algoritmos de derivação de primeira ordem

A derivada de primeira ordem, por definição, permite o cálculo da diferença entre valores pertencentes a um dado intervalo – taxa de variação. No processamento de imagem, é útil para a deteção de contornos dos objetos de interesse, uma vez que estes marcam, em geral, transições bruscas de intensidade entre pixels contíguos.

Embora haja outras, e mais completas, possibilidades, neste trabalho foi calculada a diferença central nos sentidos das linhas e das colunas da matriz (adaptando a condição da Equação 1, embora esta esteja definida para funções contínuas).

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h}$$ *Equação 1*- Equação derivada

Caso se optasse por visualizar a imagem produzida pela aplicação exclusiva da função **my_deriv** a mesma poderia interpretar-se da seguinte forma: as regiões escuras correspondem a intensidades aproximadamente constantes, e as regiões claras a zonas que marcam os limites de potenciais objetos de interesse, dado serem bastante distintos do fundo. Decorre daqui que esta função é mais eficaz quando aplicada após a imagem ter sido sujeita a remoção de ruído.

Ao invés disso, escolheu-se aplicar estes resultados de duas formas possíveis: podem ser diretamente usados na função **cont**, responsável pela definição dos contornos, ou pode ser calculada a média entre eles, resultando daí uma matriz cujos elementos máximo e mínimo serão úteis na **get_info**.

Relativamente à função **cont**, listam-se abaixo as principais adições que se realizam à ação da **my_deriv**:

- o cálculo do valor absoluto de todos os valores - para se ter uma gama de valores conhecida e evitar erros no passo seguinte;
- a binarização da matriz derivada, necessária devido à subtileza das imagens utilizadas. Assim, para minimizar a obtenção de contornos indesejados, é conveniente estabelecer um limiar, e para resultados o mais claros possível, escolher o valor de intensidade máxima (1) para os pixels da imagem dada pela derivada que estejam acima desse limiar, e a intensidade dos restantes permanece a mínima.
- No sentido de realçar os pontos correspondentes a variações notórias nos dois sentidos - linhas e colunas -, são conjugadas as duas imagens resultantes, através de operadores lógicos.

Algoritmo de apresentação de texto

Com a implementação da função **my_text**, o programa oferece ao utilizador a oportunidade de escrever o que pretende numa caixa de texto sobre a imagem.

Este algoritmo consiste na criação de um vetor com duas variáveis correspondentes às coordenadas do pixel selecionado com o cursor, através do **ginput** (o qual, dado que se trata de interação com o utilizador, está definido no sistema de processamento a discutir posteriormente). Na mesma posição, é então apresentada uma caixa de texto, na qual o utilizador insere o que pretende. Para este efeito foi utilizada a função nativa do MATLAB **insertText**. Este algoritmo prevê, ainda, que o utilizador possa escrever várias porções de texto em coordenadas diferentes, decidindo quando parar de o fazer.

Por fim, a inserção do texto é incorporada na imagem, sendo a matriz resultante guardada numa nova variável e apresentada (ex. Figura 3).

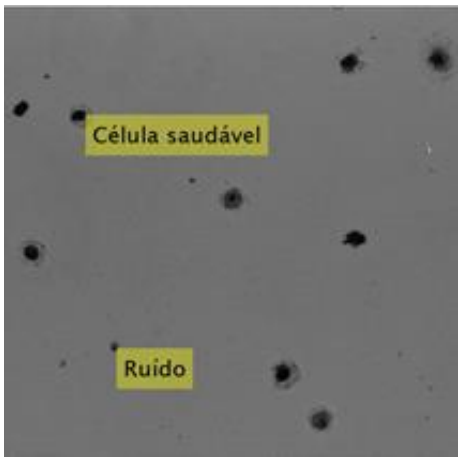


Figura 3 - Imagem com texto inserido

Algoritmo de apresentação de texto e cor

Nesta secção do programa, trabalha-se com a imagem original a cores (variável *imagergb*). A execução da função **get_info** irá apresentar sobre a mesma, dois tipos de informação.

A seleção de um ou vários pixels por parte do utilizador irá fazer surgir na imagem os valores de luminosidade do(s) mesmo(s). Como, neste caso, não se está na presença de uma imagem em escala de cinzentos, os valores a apresentar serão três - correspondentes aos tons vermelho, verde e azul. Novamente, foram utilizadas as funções nativas **ginput** e **insertText**.

Detetou-se um problema na aplicação desta função a imagens cujos blocos de interesse se localizam próximo das bordas: a caixa de texto aparecia cortada. De forma a resolver isto, alargaram-se as margens da matriz com recurso à concatenação (ex. Figura 4).

Adicionalmente, sobre coordenadas fixas, surgem os valores de mínimo e máximo da matriz da derivada (valores obtidos pela função **my_deriv**). O código para o cálculo destes situa-se antes do ciclo while para a inserção das coordenadas, de modo a só ser utilizado uma vez e não sofrer alterações, pois o texto inserido irá adulterar as intensidades da imagem.

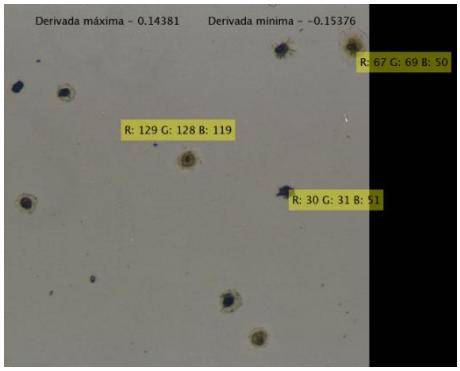


Figura 4 - Imagem com informações

Algoritmo de propriedades das células

É possível, no entanto, e talvez mais pertinente, obter outro tipo de informações, relativas às células em específico. Com este intuito, e de modo a valorizar o trabalho, elaborou-se a função **my_regionprop**, semelhante à função nativa **regionprops**.

A partir da imagem já segmentada (ver **segmen**), esta função averigua o número de regiões relevantes, e acerca delas proporciona a obtenção dos seguintes dados: coordenadas da aproximação ao centróide; área, calculada através do número de pixels; distância ao centro da célula mais próxima, calculada pela fórmula matemática da distância; aproximação dos diâmetros menor e maior da célula (dado que não é um círculo perfeito).

As informações supramencionadas relativas à imagem são, para além de concatenadas numa matriz junto com a imagem original e mostradas, guardadas num ficheiro Excel que pode posteriormente aberto.

Sistema de processamento

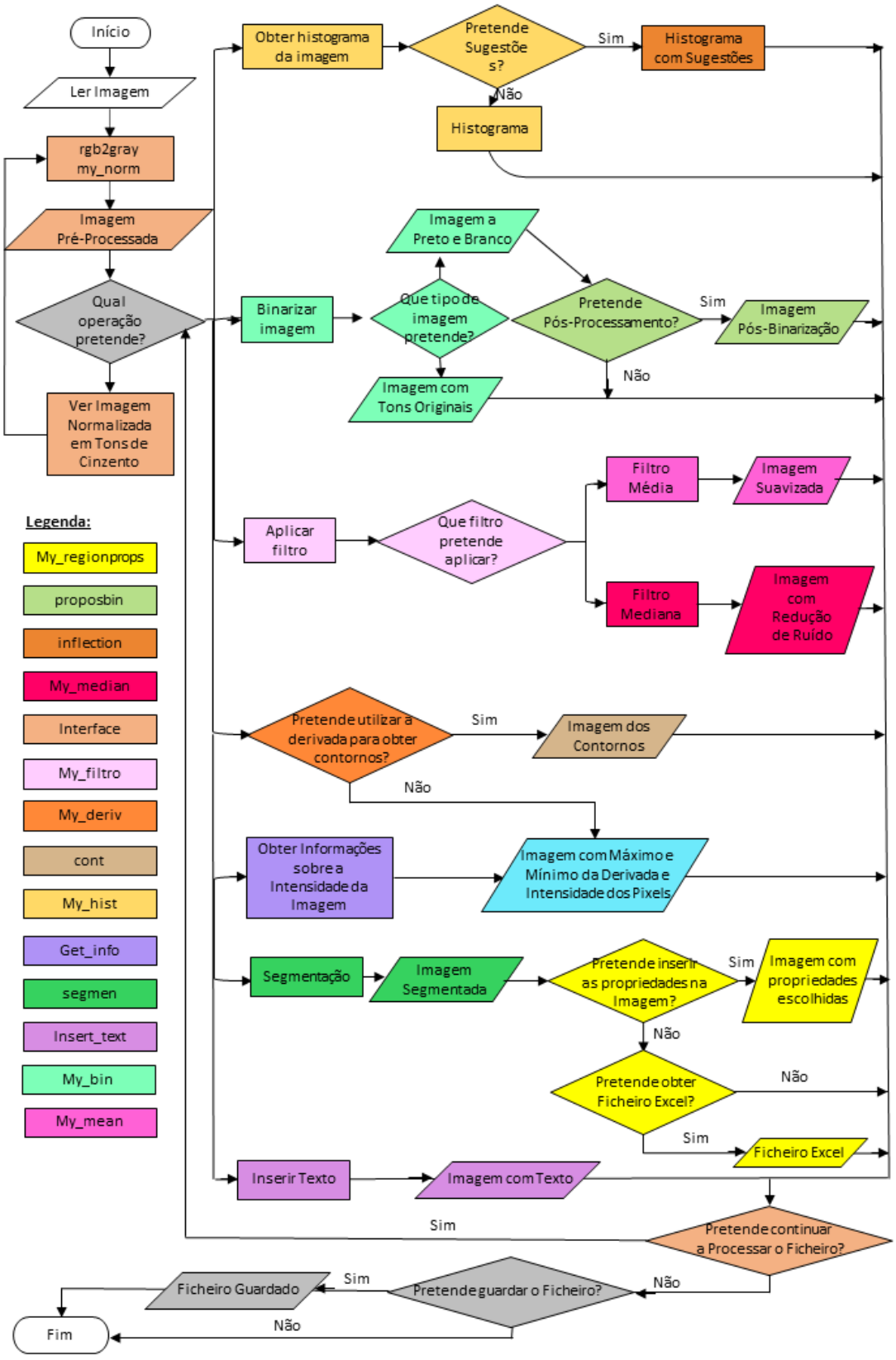
A interação do programa com o utilizador implica uma rede de perguntas e respostas que conecta todas as funções, que é indispensável para testar o cumprimento dos objetivos de todas as restantes funções do trabalho.

Existe um *script* a que se chamou **Interface** e uma função **sis**, que se relacionam no sentido de levar a cabo a interação com o utilizador, abrangendo todas as outras decisões e introduções de variáveis necessárias. **Interface** admite a inserção do nome do ficheiro, e remete para **sis**. Esta apresenta um conjunto de opções que correspondem às funções principais, desde o histograma até à segmentação. Após a aplicação de qualquer das formas de processamento da imagem, oferece-se a escolha de parar ou continuar a processar, pelo que há um ciclo while no código de **Interface** que evita a necessidade de tornar a inserir o nome do ficheiro. Caso se opte por continuar o processamento, o ficheiro utilizado é o original. Todavia, as alterações efetuadas na imagem ficam guardadas e o utilizador tem a oportunidade de guardar a imagem alterada, e caso pretenda processar esta última deve guardá-la como ficheiro e depois inserir o nome do mesmo.

Recorrendo à função nativa **listdlg**, criam-se sucessivas caixas de diálogo que permitem ao utilizador controlar a operação a realizar. Esta escolha é traduzida na alternância de variáveis que controlam blocos de instruções switch ou if, que chamam as diferentes funções.

Por sua vez, o **inputdlg** permite a introdução das variáveis necessárias. Nas funções cujos argumentos de entrada incluem dados introduzidos pelo utilizador, é importante garantir a sua concordância com o bom funcionamento do programa (por exemplo, se não são colocados números não naturais para uma vizinhança). Tentou prever-se o máximo possível destes erros, colocando-se condições de validação nas funções para as quais tal se justifica.

Fluxograma geral



Aplicação em processamento de imagem

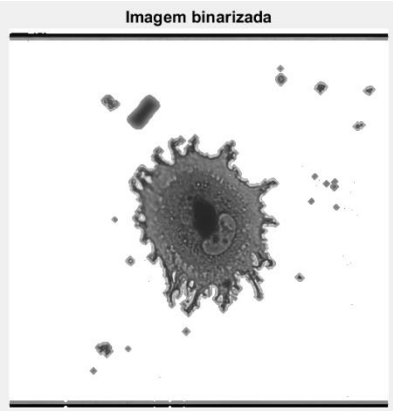


Figura 5 – celula#1.png binarizada com limites 0.12 e 0.47

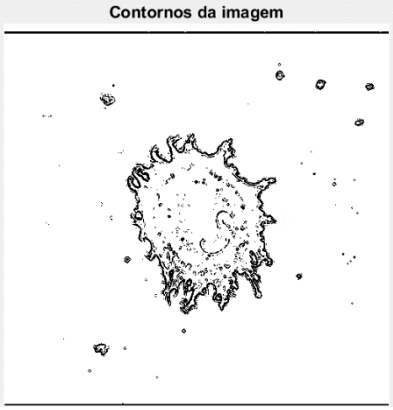


Figura 6 – Contornos de celula#1.png com limiar 0.07

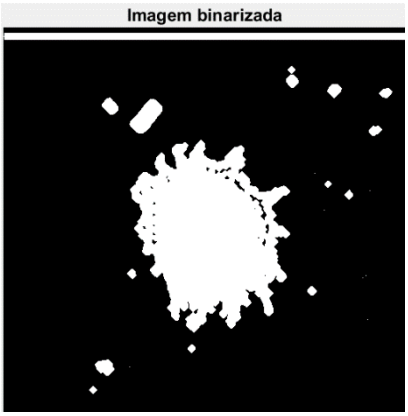


Figura 7 – celula#1.png binarizada com limites recomendados (provenientes de inflection)

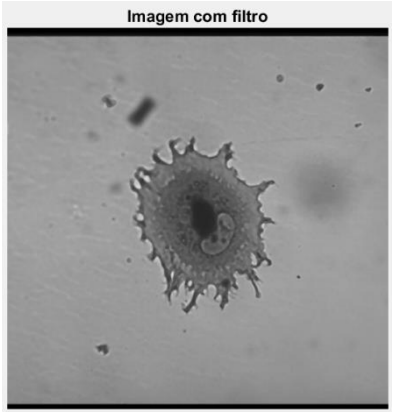


Figura 8 – celula#1.png com filtro de mediana de vizinhança 4

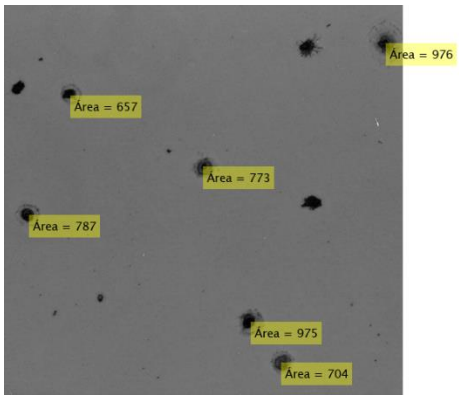


Figura 9 – celula#3.png com informação sobre área das células (limites 0.10-0.30)

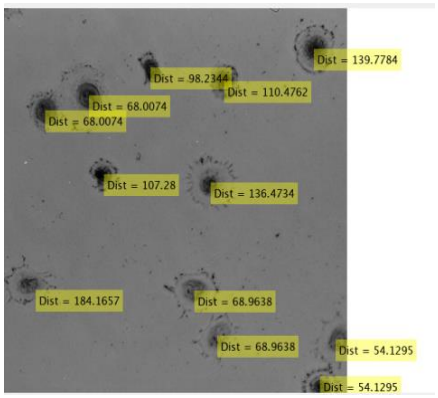


Figura 10 – celula#2.png com informação sobre distância mínima entre células (limites 0.10-0.35)

	A	B	C	D	E	F
1	X	Y	Area	DiametroMenor	DiametroMaior	DistanciaMinima
2	116	19	414	27	27	68,8839604
3	127	87	657	29	34	68,8839604
4	218	257	773	33	35	146,4137972
5	417	318	975	39	40	65
6	469	357	704	31	34	65
7	65	393	728	34	37	98,12746812
8	264	396	590	29	36	146,4137972
9	60	491	976	38	39	98,12746812

Figura 11 – Documento Excel com propriedades da celula#3.png

Conclusões

Como conclusão do presente relatório é relevante referir que foi criado um programa completo de processamento de imagem em que as funções correspondem a implementações originais, cujo objetivo principal é simplificar a análise/tratamento das imagens. Verificou-se que, para a compreensão da imagem com este fim, é vantajoso realizar segmentação semântica.

Deve-se acentuar a constante interação entre o utilizador e o programa, dado que o primeiro tem controlo sobre as alterações que pretende visualizar na imagem. Tentou-se que todo o programa fosse *user friendly*, possibilitando a uma pessoa inexperiente seguir as instruções sugeridas. De facto, o programa foi testado por pessoas alheias à sua elaboração, e revelou ser de fácil manipulação. No entanto, para uma utilização mais eficaz do programa, é conveniente não só ter algum conhecimento sobre o que representa cada imagem como também possuir alguma noção prévia acerca do tipo de alteração à imagem que pode esperar com cada operação. Isto é, se o objetivo final for obter uma imagem segmentada, deve haver algum espírito crítico que leve o utilizador a seguir uma certa sequência de passos e realizar as escolhas certas para alcançar o melhor resultado.

Deparámo-nos com várias dificuldades ao longo da realização do trabalho. Uma das primeiras prendeu-se com a interpretação do histograma, que se revelava pouco conclusivo, dada a subtileza dos tons das imagens de teste. No entanto, fizemos uso de funcionalidades do *software* que provaram ser da maior utilidade, como os *breakpoints*.

Como forma de melhorar a eficiência do código, tentou-se, sempre que possível, realizar pré-alocações de todos os vetores e matrizes de dimensão previamente conhecida, ao invés de concatenações a partir de matrizes vazias. Além disso, pretendeu-se que o código fosse o mais modular possível, tendo-se criado bastantes funções, entre as principais e as auxiliares.

Por fim, é de notar o progresso efetuado na compreensão da preparação e manipulação de imagens digitais. Destaca-se o papel do MATLAB, uma ferramenta que revelou ser da maior utilidade no processamento de imagem, e em particular como auxiliar da análise de células, cuja importância estará patente em futuros contactos que teremos com as suas aplicações em Bioengenharia.

Referências bibliográficas

1. AJC, C. J. (12 de 2020). Algoritmos de binarização, derivação e filtragem. *Guião do TP2_v1*.
2. Alfio Quarteroni, F. S. (2007). *Cálculo Científico Com MATLAB e Octave* . Springer.
3. Chapman, S. J. (s.d.). *MATLAB Programming with Applications for Engineers.* USA: Cengage Learning.
4. John N. Little, et all. (s.d.). *Mathworks*. Obtido de https://www.mathworks.com/
5. Michael Fitzpatrick, Á. L. (2013). *Computer Programming with MATLAB* .
6. https://www.ck12.org/book/ck-12-probability-and-statistics-concepts/section/4.6/
7. https://wiki.sj.ifsc.edu.br/index.php/Curso_Matlab_aplicado_ao_processamento_de_imagens
8. https://homepages.inf.ed.ac.uk/rbf/HIPR2/histgram.htm
9. https://web.fe.up.pt/~padilha/PAI/ficheiros/Cap4-ac.pdf
10. https://www.feis.unesp.br/Home/departamentos/engenhariaeletrica/pos-graduacao/273-dissertacao_patricia.pdf
11. https://www.mathworks.com/help/images/sliding-neighborhood-operations.html
12. https://homepages.inf.ed.ac.uk/rbf/HIPR2/threshld.htm
13. https://homepages.inf.ed.ac.uk/rbf/HIPR2/adpthrsh.htm#guidelines
14. https://setosa.io/ev/image-kernels/
15. https://www.mathworks.com/solutions/image-video-processing/semantic-segmentation.html
16. http://www.ic.uff.br/~aconci/suavizacao.pdf