

Processamento de Imagem

5 - Segmentação

Análise e Processamento de Imagem (M4031, M4094)

Processamento de Sinal e Imagem em Física Médica (F4012)

[Conteúdo]

click on it

André R. S. Marçal

Departamento de Matemática
Faculdade de Ciências, Univ. Porto (FCUP)
<http://www.fc.up.pt/pessoas/andre.marcal>

versão 1.0 - 17 Março 2021

Conteúdo

1	Introdução	3
2	Segmentação por <i>Thresholding</i>	4
2.1	<i>Thresholding</i> Global	4
2.1.1	Determinação automática de <i>Threshold</i>	5
2.2	<i>Thresholding</i> Local (Adaptativo)	8
3	Segmentação baseada em regiões	10
3.1	Crescimento de Regiões	10
3.2	Segmentação por Divisão de Regiões	12
3.3	Segmentação por <i>Watershedding</i>	13
3.4	Segmentação Multi-Resolução	15
4	Modelos Deformáveis	16
4.1	Contornos Activos / GVF Snakes	16
4.2	Level Sets	17
5	Segmentação de Imagens de Cor	19
6	Avaliação de segmentação	21
6.1	Avaliação por Contorno	21
6.2	Avaliação por Área	22
7	Segmentação no MATLAB	24
7.1	Segmentação com colorThresholder	24
7.2	Aplicação imageSegmenter	24
8	Referências Bibliográficas	25
9	Exercícios propostos	26
9.1	APPENDIX - Exercices (in English)	27

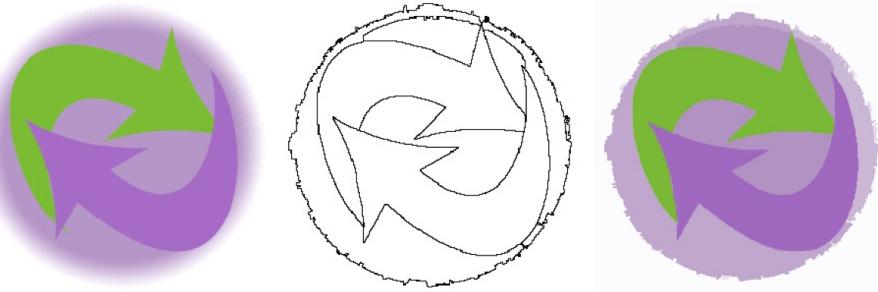


Figura 1: Modos de representação da segmentação de uma imagem colorida (esquerda): por contornos (centro); por blocos (direita).

1 Introdução

A segmentação é o processo de sub-divisão de uma imagem em regiões (segmentos), normalmente associados a alguma característica que permita auxiliar no processo de análise e extração de informação. Considerando uma imagem I segmentada em N regiões I_i , com $i = 1, 2, \dots, N$, deverão ser observadas as seguintes condições:

$$I = \bigcup_{i=1}^N I_i \quad (1)$$

$$I_i \cap I_j = \emptyset \quad \forall i, j; i \neq j \quad (2)$$

Ou seja, cada pixel pertence a uma e uma só região. A união de todos os segmentos é a imagem original.

Há 2 formas comuns de representar uma imagem segmentada. Através de contornos, muitas vezes recorrendo a dados vectoriais, que podem ser sobrepostos à imagem. Uma outra forma é por blocos, o que na verdade normalmente é feito através de uma imagem de cor indexada, onde cada valor numérico corresponde a uma região. A figura 1 mostra um exemplo de uma imagem colorida (esquerda) cujo resultado do processo de segmentação é apresentado por contornos (centro) e por blocos (direita). No caso da segmentação por blocos, escolheu-se neste caso uma tabela de cores baseada nas componentes médias de RGB de cada região, pelo que a imagem é visualmente muito parecida com a original.

O processo de segmentação pode dar origem a um grande número de regiões, mas muitas vezes pretende-se identificar apenas duas regiões - objeto e fundo. Neste caso a imagem da segmentação por blocos corresponde a uma imagem binária, com o valor 1 (ON) para o objeto (ou zona de interesse) e 0 (OFF) para o fundo (pixels sem interesse).

Numa primeira fase são consideradas imagens de 1 banda (níveis de cinzento), sendo que apenas algumas das técnicas de segmentação de imagem são extensíveis para imagem multi-banda (cor com 3 bandas ou multi-espectrais).

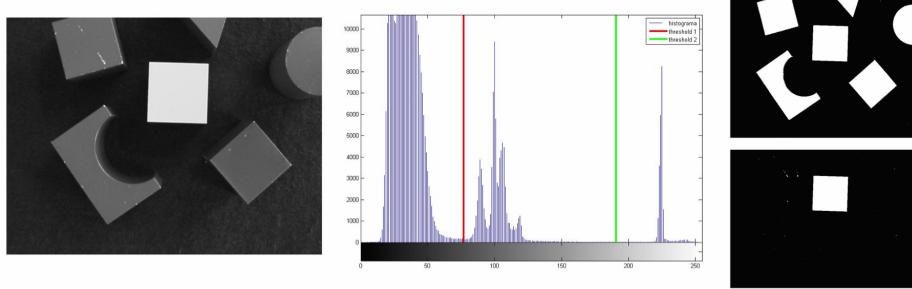


Figura 2: Exemplo de segmentação por *Thresholding Global* (T_g). Imagem original (esquerda); histograma com indicação de 2 valores para T_g (centro); imagens binárias obtidas por *Thresholding Global* para cada T_g (direita).

2 Segmentação por *Thresholding*

Thresholding é um método de segmentação bastante simples, mas que funciona bem em muitos casos. Tem por base a definição de um valor de intensidade (*Threshold*) que é usado como critério de decisão para atribuir cada pixel a um dos sub-grupos (segmentos). Na forma binária, é definido um *Threshold* (T), e a cada pixel $I(x, y)$ é atribuído um valor a ou b , de acordo com (3), onde $S(x, y)$ é a imagem segmentada. Normalmente considera-se $a = 0$ e $b = 1$, embora possam ser usados outros valores. A este processo pode chamar-se binarização.

$$S(x, y) = \begin{cases} a & \text{se } I(x, y) \leq T \\ b & \text{se } I(x, y) > T \end{cases} \quad (3)$$

O método de *Thresholding* pode também ser usado para dividir a imagem em mais do que 2 sub-grupos (segmentos), o que se chama habitualmente *Multi-Thresholding*. Para N sub-grupos ($N > 2$) é necessário definir $N - 1$ *Thresholds*, satisfazendo a condição $T_1 < T_2 < \dots < T_{N-1}$. A imagem segmentada $S(x, y)$ toma valores $\{a, b, \dots, z\}$, de acordo com (4) .

$$S(x, y) = \begin{cases} a & \text{se } I(x, y) \leq T_1 \\ b & \text{se } T_1 < I(x, y) \leq T_2 \\ \dots & \\ z & \text{se } I(x, y) > T_{N-1} \end{cases} \quad (4)$$

2.1 *Thresholding Global*

No caso binário é apenas necessário definir um valor de T , que pode ser escolhido a partir do histograma da imagem. A figura 2 mostra um exemplo de uma imagem de tons de cinzento (esquerda) e o respectivo histograma (centro). Pode observar-se que há um grande aglomerado de pixels com valores muito baixos (escuros), um aglomerado de valores intermédios e um outro de valores altos (claros). No histograma são indicados 2 valores possíveis para T , assinalados a vermelho e verde. A figura 2 (direita) mostra as imagens binárias obtidas por *Thresholding* com cada valor de T . Com o T maior (verde), apenas os pixels de tonalidade muito clara aparecem como objeto (ON) na imagem segmentada.

As imagens binárias no exemplo da figura 2 foram obtidas por *Thresholding* Global. Ou seja, cada pixel é tratado de forma igual em toda a imagem, com base apenas no seu valor I , sendo usado o parâmetro T em (3) como critério de decisão. Neste caso os valores de T foram atribuídos manualmente, $T = 80$ (vermelho) e $T = 190$ (verde), na escala de 0 a 255 (8 bits sem sinal).

Há essencialmente 3 possibilidades para a escolha de T : (1) manualmente; (2) interactivamente (forma semi-automática); (3) automaticamente. O processo interactivo conciste em ajustar o parâmetro por observação do resultado. Os métodos automáticos serão em princípio os mais interessantes, uma vez que não exigem intervenção de um operador, sendo o parâmetro T obtido por um processo de optimização.

2.1.1 Determinação automática de *Threshold*

Há várias métodos para calcular o valor óptimo de T (*Threshold*) de forma automática, sem intervenção de um operador. Uma possibilidade é usar um método iterativo, como o procedimento indicado em seguida:

1. Selecionar estimativa inicial T_0 para o *Threshold* T ; $i = 1$; $T_i = T_0$
2. Segmentar a imagem com valor de corte T_i
3. Calcular as médias m_1 e m_2 da intensidade (I) dos pixels atribuídos às regiões 1 ($I \leq T$) e 2 ($I > T$); $i++$
4. Calcular um novo valor de corte $T_i = (m_1 + m_2)/2$
5. Se $|T_i - T_{i-1}| > \Delta m$ voltar para 2. Caso contrário terminar; $T = T_i$.

A estimativa inicial T_0 poderá ser a média ou mediana da imagem I , o ponto médio do intervalo de valores utilizados, ou simplesmente o ponto médio do intervalo de valores possíveis. Para dados de variável discreta (ex. 8-bits), Δm pode por exemplo ser 0.5, que é inferior ao menor incremento possível (1).

Uma outra possibilidade é usar o método de Otsu [Otsu] que procura o valor de T que minimiza a variância intra-classes, ou variância interna (σ_i^2). O valor de σ_i^2 é calculado como uma média ponderada das variâncias das 2 classes (1 e 2), de acordo com (5), onde ω_1 e ω_2 são as probabilidades dum pixel pertencer às classes 1 e 2, para um dado T , e σ_1^2 e σ_2^2 as variâncias de cada classe.

$$\sigma_i^2(T) = \omega_1(T)\sigma_1^2(T) + \omega_2(T)\sigma_2^2(T) \quad (5)$$

Neste caso, uma vez que se tem apenas 2 classes, a minimização de variância intra-classes é equivalente à maximização da variância inter-classes, ou variância externa (σ_e^2). O valor de σ_e^2 pode ser calculado através de (6), onde σ^2 é a variância de toda a imagem.

$$\sigma_e^2(T) = \sigma^2 - \sigma_i^2(T) \quad (6)$$

Quer o método iterativo quer o método de Otsu são particularmente eficientes em imagens com histograma bimodal. Nesses casos os valores *Threshold* T previstos pelos 2 métodos são iguais ou muito próximos, com as imagens binárias resultantes praticamente indistinguiveis. A figura 3 mostra alguns exemplos de segmentação / binarização de imagens de tons de cinzento (1 banda), através

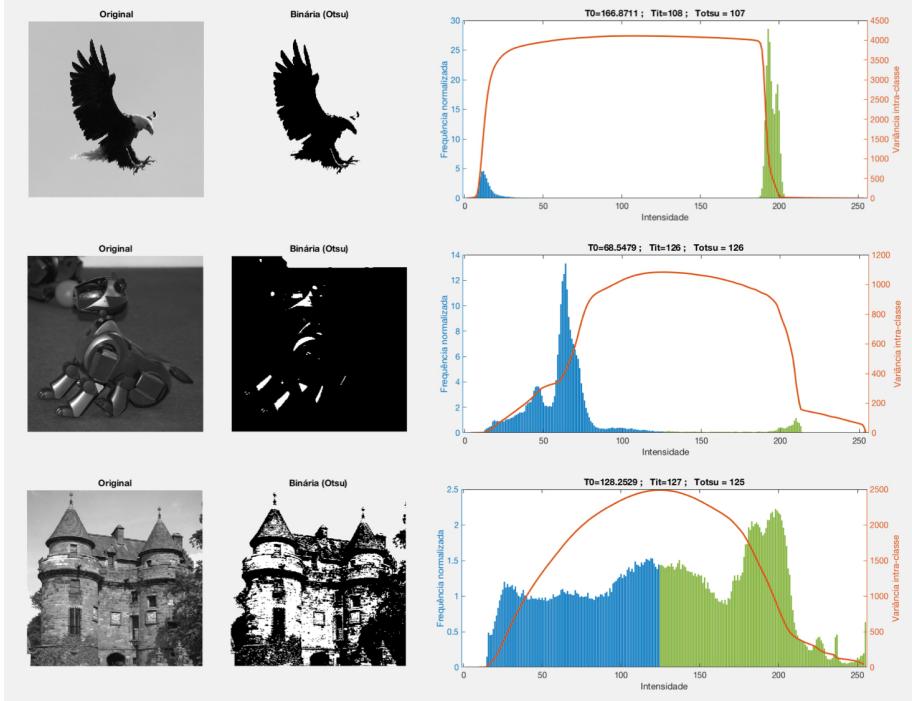


Figura 3: Exemplo de segmentação por *Thresholding Global* (T_g). Imagem original (esquerda); histograma com indicação de 2 valores para T_g (centro); imagens binárias obtidas por *Thresholding Global* para cada T_g (direita).

de *Thresholding Global*. Em cada linha da figura 3 é apresentada a imagem original (esquerda), o seu histograma (direita), e a versão binária com T obtido de forma automática pelo método de Otsu (centro). O histograma tem as barras em 2 cores - azul e verde, correspondendo às 2 classes ou segmentos, que são apresentados com 0 e 1 na imagem binária. Os valores são apresentados como frequências normalizadas, onde 1 corresponde ao valor do histograma uniforme. É igualmente apresentada sobre o histograma o gráfico de $\sigma_e^2(T)$ (a laranja), que se pretende maximizar, e os seguintes valores de T (*Threshold*): T_0 (estimativa inicial, mediana de I), T_{it} (método iterativo) T_{otsu} (método de Otsu).

No primeiro caso (Aguia), o histograma é claramente bimodal. O gráfico de $\sigma_e^2(T)$ tem um patamar largo de valores muito próximos do máximo, pelo que pequenas variações de T nesta zona tem efeitos imperceptíveis na imagem binária resultante. O valor de T que maximiza $\sigma_e^2(T)$ é 107, e o valor previsto pelo método iterativo é 108, pelo que as imagens binárias produzidas pelos 2 métodos são praticamente iguais. No segundo caso (Robot), o histograma também é bimodal, mas há alguns pixels com valores intermédios, entre as 2 zonas de maior densidade. Nesta imagem, os 2 métodos previram o mesmo valor de T - 126. No terceiro caso (Palácio de Falkland), a imagem tem um histograma muito próximo de uniforme. Os valores de T previstos foram 125 (Otsu) e 127 (Iterativo), o que irá dar origem a imagens binárias diferentes. No entanto, dadas as características do histograma, nenhum destes métodos será

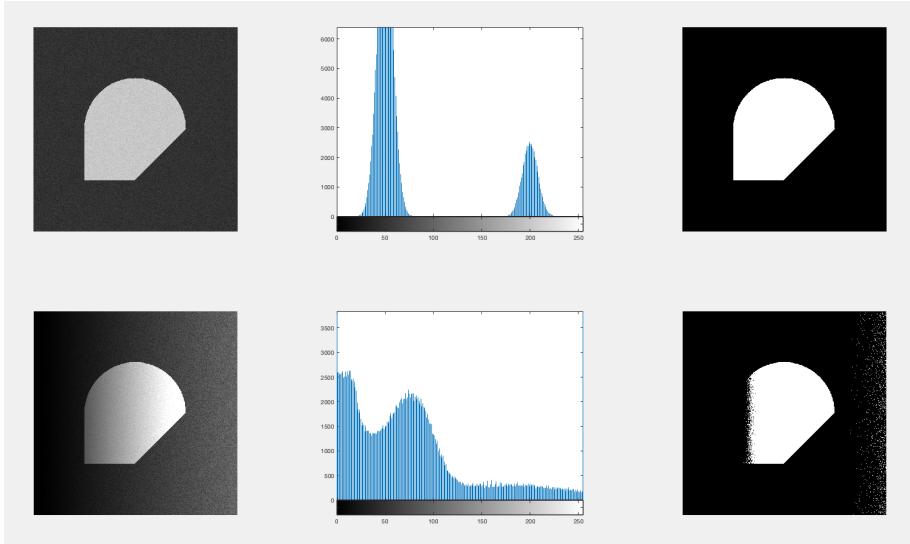


Figura 4: Exemplo para ilustrar o impacto da iluminação na segmentação de um objecto por *Thresholding Global*: caso I - iluminação uniforme (em cima); caso II - iluminação variável (em baixo). Imagem original (esquerda); correspondente histograma (centro); imagem binária (direita).

particularmente adequado para escolher um valor de corte para a binarização.

A segmentação por *Thresholding Global* poderá não ser a mais adequada para imagens com as características da imagem de teste 'Falkland' (figura 3, em baixo). Na verdade os métodos de Otsu e iterativo apenas são eficientes na determinação do *Threshold* T_g em imagens com histograma bimodal.

A figura 4 mostra um outro exemplo de segmentação por *Thresholding Global*, onde são identificadas limitações relacionadas com iluminação variável. Consideram-se 2 versões de uma imagem de um objecto claro com fundo escuro - uma com iluminação uniforme (caso I, linha de cima) e outra com iluminação variável (caso II, linha de baixo). Como se pode observar nas imagens originais (esquerda) e nos histogramas (centro), no primeiro caso os pixels do objecto e do fundo tem intensidades claramente diferentes, resultando num histograma bi-modal com 2 grupos de valores densamente ocupados. No segundo caso, a variação da iluminação ao longo das colunas da imagem faz com que os dois grupos (objecto, fundo) se sobreponham e em consequência não seja claro que algum valor de T_g permita segmentar de forma satisfatória o objecto. Na figura 4 são igualmente apresentadas as imagens binárias obtidas por *Thresholding Global* (direita), tendo os valor de T_g sido obtidos pelo método de Otsu ($T_{otsu} = 125$ no primeiro caso e $T_{otsu} = 118$ no segundo). Na imagem binária obtida no caso II, há falsos positivos (pixels de fundo identificados como objecto, à direita na imagem), e falsos negativos (pixels do lado esquerdo do objecto não identificados). Aumentando o valor de T_g tem como efeito uma redução de falsos negativos mas também um aumento de falsos positivos. Por outro lado, diminuindo T_g resulta numa redução de falsos positivos e num aumento de falsos negativos. Não há nenhum valor T_g que segmente o objecto da forma pretendida, como no caso I.

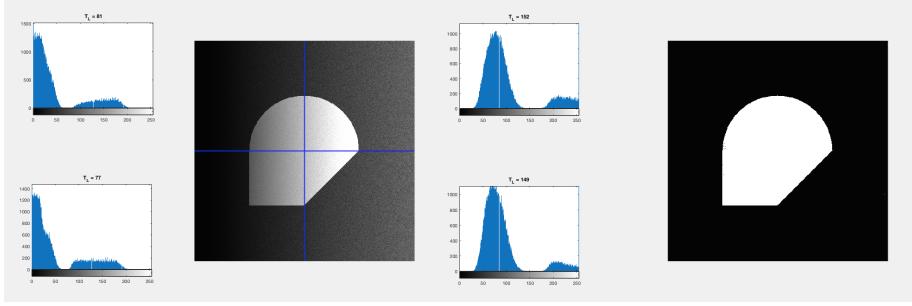


Figura 5: Imagem original de um objecto com iluminação variável (figura 4, caso II) e histogramas das 4 sub-imagens identificadas (esquerda); imagem binária resultante de *Thresholding Local* com $N=2$ (direita).

2.2 Thresholding Local (Adaptativo)

A técnica de *Thresholding Local* (ou Adaptativo) é uma alternativa para imagens onde não é possível encontrar um valor satisfatório para T_g (*Threshold Global*), como no caso II da figura 4. Há várias formas de implementar segmentação (ou binarização) por *Thresholding Local*. Uma delas consiste em dividir a imagem original em $N \times N$ sub-imagens, que são processadas (binarizadas) separadamente. Para auxiliar a descrição do método, consideram-se exemplos com $N=2$ (figura 5) e $N=5$ (figura 6) usando a imagem sintética do objecto claro sobre fundo escuro com iluminação variável (figura 4, caso II).

Na figura 5 estão assinaladas a azul as linhas de divisão da imagem original em 4 sub-imagens (2×2), e são apresentados os histogramas de cada sub-imagem. Pode verificar-se que os histogramas são todos bi-modais, mas que as sub-imagens do lado direito tem intensidades mais altas do que as do lado esquerdo. Os valores de T_L (*Threshold Local*) são calculados por um método automático [2.1.1](#) para cada sub-imagem, tendo neste caso sido usado o método de Otsu. Os valores de T_L obtidos foram os seguintes: 81 (sup.esq.), 77 (inf.esq.), 160 (sup.dir.) e 150 (inf.dir.). Cada sub-imagem é binarizada com o seu T_L , e em seguida as 4 imagens binárias são reagrupadas para formar a imagem segmentada final (figura 5, direita). Comparando este resultado com o obtido por *Thresholding Global* (figura 4), verifica-se que o *Thresholding Local* foi mais eficaz a segmentar o objecto, havendo no entanto ainda alguns pixels do fundo, no lado direito da imagem, a que foi erradamente atribuído o valor 1 (ON).

Na figura 6 estão assinaladas a amarelo as linhas de divisão da imagem original em 25 sub-imagens (5×5), e a azul claro as 4 sub-imagens cujos os histogramas são apresentados. Focando na primeira sub-imagem (canto superior esquerdo), verifica-se que todos os pixels tem valores muito baixos, não sendo adequado fazer *Thresholding*, uma vez que a variância é muito baixa. O bloco (sub-imagem) é considerado uniforme, sendo neste caso atribuído o valor 0 a todos os seus pixels. O mesmo acontece para a sub-imagem destacada no canto superior direito. Por outro lado, a sub-imagem destacada na parte inferior esquerda da imagem apresenta um histograma bimodal, pelo que se aplica *Thresholding* com T_L obtido pelo método de Otsu ($T_L = 63$). A sub-imagem no centro também é considerada homogénea, mas com o valor 1 atribuído a todos

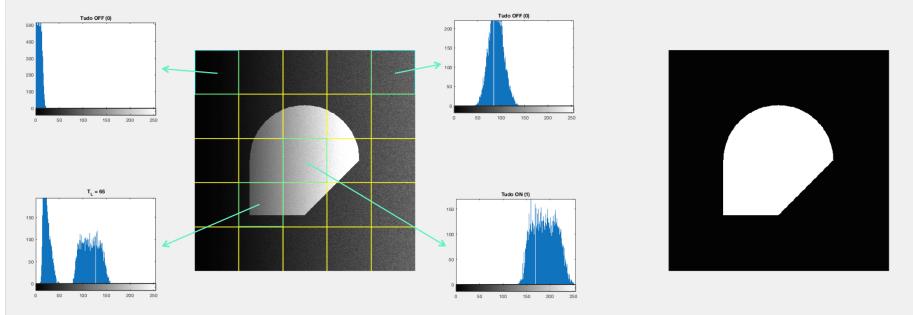


Figura 6: Imagem original de um objecto com iluminação variável e histogramas de 4 sub-imagens selecionadas (esquerda); imagem binária resultante de *Thresholding* Local com $N=5$ (direita).

os seus pixels. A imagem segmentada final resulta do reagrupamento das 25 sub-imagens binárias, sendo neste caso o resultado obtido muito bom (figura 6, direita). Há no entanto pelo menos 2 aspectos críticos que tem de ser definidos para a implementação automática deste tipo de *Thresholding* local. (1) Qual o critério para decidir se um bloco é ou não uniforme? (2) Qual o valor a atribuir a um bloco considerado uniforme, 0 ou 1? Para além disso, a qualidade do resultado final depende da escolha do valor do parâmetro N , que deveria em princípio ser feita de forma automática...

Relativamente ao primeiro ponto (critério de homogeneidade), pode ser usada a variância da sub-imagem, comparando com um valor de referência fixo ou eventualmente ajustado pela variância total da imagem e/ou valor de N . É no entanto difícil definir um critério robusto que seja eficiente em todo o tipo de imagens que se pretendam segmentar. O segundo ponto é mais fácil de resolver, embora não de forma infalível. Pode usar-se como critério de decisão sobre o valor a atribuir a um bloco uniforme (0 ou 1) a comparação entre a sua intensidade média e T_G (*Threshold Global*). A escolha do valor N óptimo é no entanto difícil de implementar, em particular sem recorrer a dados de referência.

O procedimento para implementação da segmentação por *Thresholding* local descrita, pode ser resumido da seguinte forma:

1. Escolher um valor para N .
2. Dividir a imgem em $N \times N$ sub-imagens (blocos).
3. Processar cada bloco, criando uma sub-imagem binária.
 - Verificar se o bloco é uniforme.
 - Se SIM, decidir o valor (0 ou 1) a atribuir a todo o bloco.
 - Se NÃO, calcular T_L (ex. com Otsu) e aplicar *Thresholding* ao bloco.
4. Juntar as $N \times N$ sub-imagens binárias, para criar a imagem final

Para além dos pontos identificados anteriormente, pode haver outras dificuldades de implementação deste procedimento, por exemplo relacionadas com o tamanho das sub-imagens (que podem ter de ser diferentes). Convém referir que há outras formas de implementar *Thresholding* local para segmentação.

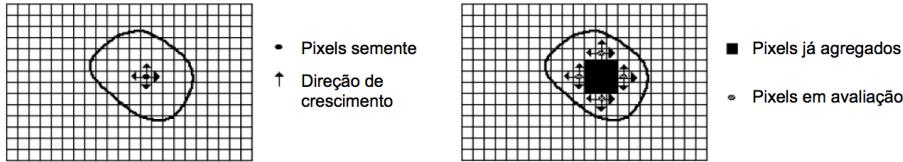


Figura 7: Esquema de fases de processamento inicial (esquerda) e intermédia (direita) do método de segmentação por crescimento de regiões.

3 Segmentação baseada em regiões

A segmentação baseada em regiões consiste na partição de uma imagem I em regiões I_1, I_2, \dots, I_N que, para além de satisfazer as propriedades 1 e 2, deverão também satisfazer condições relacionadas com a coerência e integridade espacial de cada região. As condições, que estão listadas em baixo, consideram um critério de vizinhança (\mathcal{V}) e uma propriedade (\mathcal{P}).

1. A imagem I é a reunião de todas as regiões (1)
2. A interseção de 2 regiões é um conjunto vazio (2)
3. I_i é uma região ligada, ou seja, $\mathcal{V}(I_i) = 1 \quad \forall i = 1, 2, \dots, N$
4. $\mathcal{P}(I_i) = 1 \quad \forall i = 1, 2, \dots, N$
5. $\mathcal{P}(I_i \cup I_j) = 0 \quad \forall i, j = 1, 2, \dots, N; i \neq j$

O operador \mathcal{V} aplicada à região I_i devolve 1 (verdade) se e só se houver um caminho entre quaisquer 2 pixels pertencentes à região. Caso contrário devolve 0 (falso). Normalmente o critério de vizinhança utilizado é os 4-vizinhos (o pixel e os pixels adjacentes N,S,E,O) ou os 8-vizinhos (quadrado de 3×3 pixels).

O operador \mathcal{P} aplicada à região I_i devolve 1 (verdade) se e só se todos os pixels de I_i satisfizerem a propriedade, que pode estar relacionada com a intensidade, cor, textura, forma, etc..

3.1 Crescimento de Regiões

A segmentação por Crescimento de Regiões começa com um conjunto de pontos semente (sub-regiões iniciais), que vão crescendo através de um processo iterativo, utilizando uma relação de vizinhança \mathcal{V} e uma propriedade \mathcal{P} . O processo é ilustrado na figura 7, considerando 4-vizinhos para \mathcal{V} . Inicialmente a região tem apenas 1 pixel - a semente (figura 7, esquerda). O passo seguinte é identificar os pixels candidatos, que neste caso são os 4 vizinhos do pixel semente, e verificar para cada um deles se satisfazem a propriedade \mathcal{P} . Os que satisfizerem são agregados à região e os seus vizinhos que ainda não pertençam à região são identificados como pixels candidatos. A figura 7 (direita) mostra uma fase intermédia do processo iterativo, com alguns pixels agregados e outros em avaliação, e o contorno da região no final do processamento. Na região final, os vizinhos de cada pixel agregado que não pertencem à região não satisfazem \mathcal{P} .

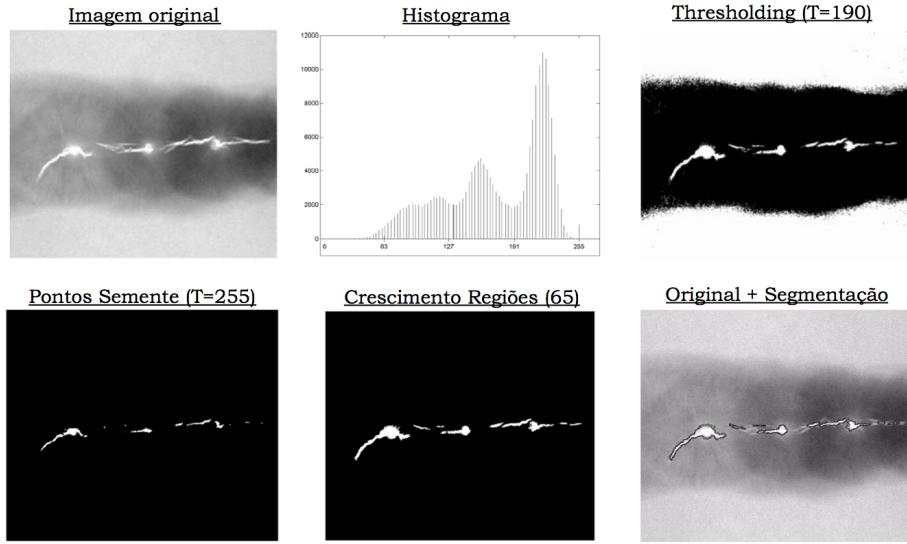


Figura 8: Segmentação de imagem industrial de soldadura para identificação de zonas com defeito. Em cima: imagem original de 8 bits (esquerda); histograma (centro); segmentação por *Thresholding* Global com $T_G=190$ (direita). Em baixo: pixels semente (esquerda), segmentação por Crescimento de Regiões na forma binária (centro); e com contornos sobrepostos à imagem original (direita).

A figura 8 mostra um exemplo de segmentação de uma imagem industrial de soldadura. O objectivo é identificar (segmentar) as zonas com defeito, que são os pixels mais claros na faixa central da imagem [GonzalezM]. Na linha de cima da figura 8 é apresentada a imagem original de 8 bits (esquerda), o seu histograma (centro) e a imagem binária obtida por *Thresholding* Global com $T_G=190$ (direita). Este resultado não é satisfatório, uma vez que para além dos pixels claros relacionados com os defeitos, há um grande número de pixels de fundo que foram igualmente identificados (valor 1 na imagem binária).

A segmentação da imagem de soldadura por Crescimento de Regiões foi feita usando 8-vizinhos (quadrado de 3×3 pixels) para \mathcal{V} , e a intensidade ser superior a 190 ($I > 190$) para \mathcal{P} . Inicialmente foi criada uma imagem com pixels semente, através da aplicação de *Thresholding* Global com $T_G=255$. Em seguida foi aplicado o processo de Crescimento de Regiões com esta imagem semente, a vizinhança \mathcal{V} e propriedade \mathcal{P} . Na linha de baixo da figura 8 é apresentada a imagem binária com os pixels semente (esquerda), e o resultado segmentação por Crescimento de Regiões na forma binária (centro) e através de contornos sobrepostos à imagem original (direita). Como se pode ver a segmentação corresponde à zona de defeitos de soldadura, com uma excelente correspondência com o que se pretendia identificar na imagem original.

Neste caso o resultado é uma imagem binária com apenas um objecto (pixels com valor 1) e fundo (0). No entanto, a segmentação por Crescimento de Regiões pode ser feita para múltiplos objectos, usando diferentes sementes. A imagem segmentada resultante terá vários objectos (1,2,...,N) e eventualmente poderá também ter pixels de fundo (0), que não pertençam a nenhum dos objectos.

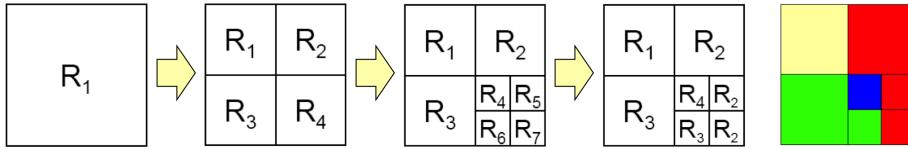


Figura 9: Exemplo esquemático do processo de segmentação de uma imagem por Divisão de Regiões.

3.2 Segmentação por Divisão de Regiões

A segmentação por Divisão de Regiões é de certa forma uma abordagem contrária à de crescimento de regiões, uma vez que se começa com toda a imagem e vai-se dividindo em regiões menores até que não se justifique continuar a dividir. Na verdade o processo iterativo conciste em dividir e agrregar (*split and merge*), embora haja outras variantes. O procedimento tem por base uma propriedade (\mathcal{P}) ou critério de homogeneidade. Uma região R_i é considerada adequada se satisfizer essa propriedade, caso contrário divide-se a região em sub-regiões. Por outro lado, regiões adjacentes que tenham semelhança pela propriedade \mathcal{P} ou critério de homogeneidade deverão ser agregadas. Uma das formas de divisão da imagem e/ou regiões é em quadrantes, o que dá origem a uma estrutura *quadtree*. A segmentação por Divisão de Regiões pode ser implementada através de duas fases: 1-dividir ; 2- agrregar. Em seguida uma breve descrição dos procedimentos para cada fase, sendo a região inicial (R_1) toda a imagem.

1. Para cada região R_i , verificar se $\mathcal{P}(R_i) = 1$
 - Se SIM, manter a região R_i .
 - Se NÃO, dividir a região em 4 sub-regiões (quadrantes).
2. Para cada par de regiões adjacentes R_i, R_j , verificar se $\mathcal{P}(R_i \cup R_j) = 1$
 - Se SIM, agrregar as 2 regiões (colocar a etiqueta i na região R_j).
 - Se NÃO, manter as regiões R_i e R_j .

O ponto 1 é repetido para todas as regiões até que não haja mais divisões, sendo atribuído um novo identificador (i) a cada nova região. Em seguida o ponto 2 é executado até que não seja possível agrregar mais regiões.

O processo é ilustrado através de um exemplo na figura 9. Inicialmente há apenas uma região (R_1) que não satisfaz \mathcal{P} , pelo que é dividida em 4 regiões. Uma delas (a do canto superior esquerdo) mantém o identificador 1, e as restantes tomam novos valores (2,3,4). Analisando estas 4 regiões, dividiu-se a região R_4 em 4 sub-regiões e manteve-se as regiões R_1 , R_2 e R_3 inalteradas. Nesta configuração, todas as 7 regiões satisfazem \mathcal{P} pelo que o processo de divisão termina (fase 1). Em seguida avança-se para o processo de agregação (fase 2), verificando-se todos os pares de regiões adjacentes. No exemplo da figura 9, a região 6 foi agregada com a 3, e as regiões 5 e 7 com a 2 (primeiro a região 5 e depois também a 7). No final, a imagem tem apenas 4 regiões (ou segmentos), que são apresentados com valores numéricos de 1 a 4 ou através de cores.

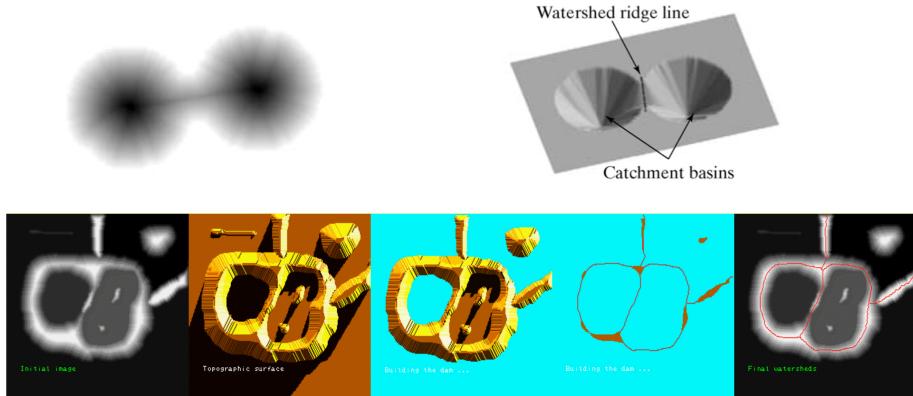


Figura 10: Segmentação por *Watersheding*. Em cima: imagem de níveis de cinzento (esquerda) e visão 3D do modelo de elevação do terreno correspondente (direita). Em baixo - exemplo de processamento (a partir da esquerda): imagem original; modelo de elevação do terreno; fases inicial e quase final do processamento (água a azul); *Watersheds* sobre imagem (vermelho) [Beucher].

3.3 Segmentação por *Watersheding*

A segmentação por *Watersheding* tem uma inspiração geográfica, relacionando-se em particular com a topografia e hidrografia. Uma imagem de uma banda é vista como uma superfície topográfica, sendo a intensidade (nível de cinzento) correspondente à elevação do terreno. O processo de segmentação consiste em fazer elevar o nível de água, como aconteceria se estivesse a chover. As zonas inundadas dão origem a regiões que se vão juntando à medida que o nível da água sobe. As cristas das elevações que separam a drenagem entre as bacias de captação de água são chamadas tergos ou cumeadas (*Watersheds*). A figura 10 (em cima) ilustra a estratégia usada com uma imagem de tons de cinzento (esquerda) e a correspondente representação da imagem como modelo de elevação do terreno (direita). Neste caso há 2 bacias de captação com uma linha de separação entre elas (*Watershed*). À medida que o nível da água sobe, cada um dos lagos vai aumentando até que se fundem.

Na figura 10 (em baixo) é apresentado um exemplo de segmentação por *Watersheding*. A começar da esquerda para a direita está a imagem original e o modelo de elevação do terreno, apresentado em 2D mas com uma escolha de cores que favorece a percepção do relevo. As duas imagens seguintes mostram uma fase inicial da inundação com água (a azul claro), e uma fase muito próxima do final. Por último são apresentadas linhas de separação das bacias, *Watersheds* (a vermelho), sobrepostas à imagem inicial [Beucher]. À medida que se vai aumentando o nível da água, as áreas não inundadas vão sendo cada vez menores, até que se fica apenas com as linhas de separação entre as bacias de captação. A transformada *Watershed* permite identificar estas linhas e as correspondentes regiões (bacias de captação).

Há várias formas de definir o modelo de terreno que serve de base ao processo de segmentação por *Watersheding*, incluindo usar a imagem diretamente. No entanto esta abordagem não é normalmente satisfatória. Uma possibilidade

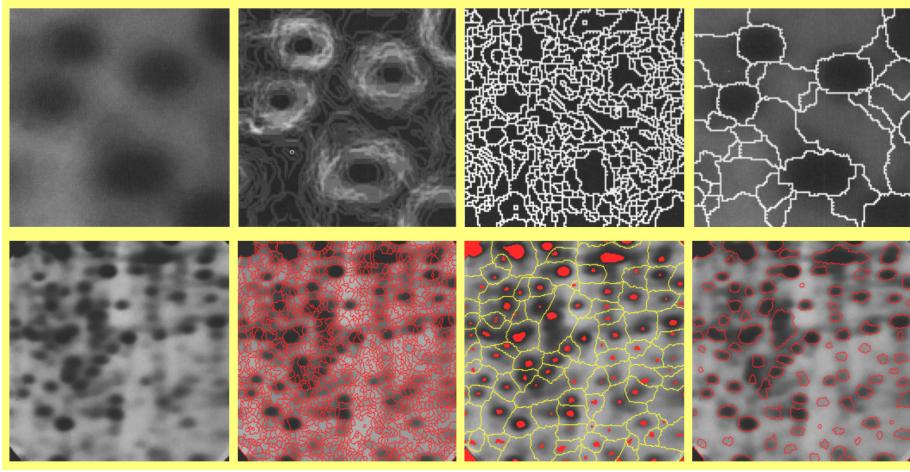


Figura 11: Exemplos de segmentação por *Watersheding* [Beucher]. Em cima (a partir da esquerda): imagem original (1); gradiente (2); segmentação a partir do gradiente (3) e do gradiente suavizado (4). Em baixo: original (1); segmentação a partir do gradiente (2); e com base em marcadores internos (3) e externos (4).

particularmente útil para imagens binárias é usar uma imagem de distâncias, onde a cada pixel é atribuído um valor que corresponde à distância para o pixel ON mais próximo. Para imagens de níveis de cinzento, uma forma habitual de definir o modelo de terreno é recorrer a uma imagem de contornos, por exemplo o gradiente.

Na figura 11 são apresentados dois exemplos de segmentação por *Watersheding* [Beucher], sendo as imagens originais (esquerda) de níveis de cinzento a 8 bits (1 banda). No primeiro exemplo (em cima), foi usado o gradiente (2^a imagem a partir da esquerda) como input da transformada *Watershed*. O resultado (3^a imagem) é uma imagem com segmentação claramente excessiva. Isto é uma característica habitual desta abordagem, uma vez que o gradiente é muito sensível, detectando pequenas variações de intensidade que normalmente não são relevantes. Numa segunda tentativa foi usada uma imagem do gradiente suavizado, o que deu origem a uma imagem segmentada com um número de regiões bastante menor (direita), mais de acordo com o que seria desejável.

No segundo exemplo (figura 11, em baixo), a imagem segmentada por *Watersheding* a partir do gradiente também apresenta uma segmentação excessiva (2^a imagem a partir da esquerda). Neste caso usaram-se marcadores de 2 tipos para evitar sobre-segmentação: marcadores internos (associados a objectos) e marcadores externos (associados ao fundo). As tarefas de pré-processamento para se identificar os marcadores podem incluir *Tresholding*, filtros espaciais, operações morfológicas, etc, não sendo discutidas aqui. O escolha de marcadores condiciona fortemente os resultados obtidos. Neste exemplo, com marcadores internos (3^a imagem a partir da esquerda) as linhas que separam as regiões são apresentadas a amarelo e os marcadores a vermelho. As regiões são de tamanho médio. Com marcadores externos (direita) as regiões são bastante menores, havendo uma região de 'fundo' que cobre mais de metade da área da imagem.

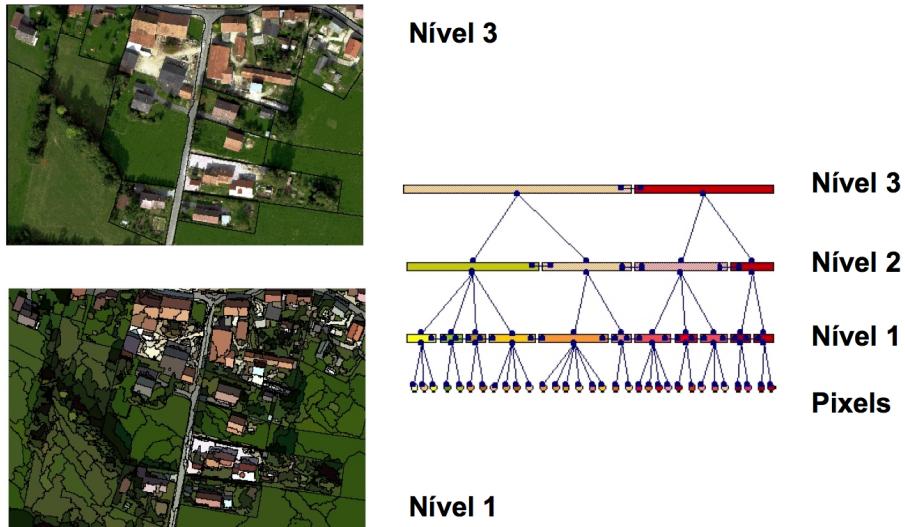


Figura 12: Exemplo de segmentação Multi-Resolução: detalhe da estrutura hierárquica, com 3 níveis de segmentação (direita); resultado da segmentação apresentado como contornos sobrepostos à imagem original (esquerda), para o nível 3 (em cima) e para o nível 1 (em baixo).

3.4 Segmentação Multi-Resolução

A segmentação Multi-Resolução é uma forma interessante de apresentação e observação de resultados, sendo particularmente útil em casos onde o processo de segmentação dá origem a um elevado número de regiões. A ideia é obter não apenas um resultado da segmentação da imagem, mas sim vários níveis, estruturados hierarquicamente. O utilizador pode depois escolher o nível mais adequado por observação da imagem e das regiões associadas a cada nível.

A figura 12 mostra um exemplo de segmentação Multi-Resolução de uma imagem aérea. Neste caso a estrutura hierárquica tem apenas 3 níveis: o nível 1 é o primeiro acima do nível do pixel individual, tendo por isso muitas regiões relativamente pequenas. Cada região do nível 2 é composta por várias regiões do nível 1, e as regiões do nível 3 são obtidas pela agregação de regiões do nível 2. Na figura 12 (direita) é apresentado uma versão simplificada da estrutura hierárquica. Na figura são igualmente apresentados resultados da segmentação a nos níveis 1 e 3, através dos contornos das regiões sobrepostos à imagem original. Como se pode observar, a segmentação nível 1 parece ser excessiva em algumas zonas da imagem, por exemplo nas parcelas agrícolas na parte inferior esquerda. No entanto poderá ser adequada nas zonas urbanizadas na parte superior direita da imagem. Por outro lado, a segmentação nível 3 poderá ser demasiado simplificada nestas zonas, mas adequada para as parcelas agrícolas. Ou seja, o nível hierárquico mais relevante pode não ser o mesmo para todas as zonas da imagem. O operador pode escolher o nível de acordo com as características das zonas de interesse e pela observação dos resultados da segmentação Multi-Resolução nos vários níveis.

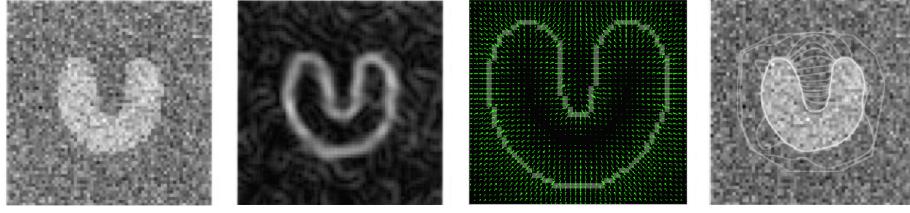


Figura 13: Segmentação por Gradient Vector Flow (GVF) Snakes (a partir da esquerda): imagem original; gradiente; campo de forças externas; curva em várias fases do processamento iterativo sobreposta à imagem original [Xu].

4 Modelos Deformáveis

Os Modelos Deformáveis usados em segmentação de imagem podem dividir-se em modelos paramétricos (Contornos Activos) e não paramétricos (modelos geométricos). A segmentação por Modelos Deformáveis baseia-se na evolução de uma curva que, sob influência de determinadas forças, altera a sua forma aproximando-se progressivamente dos limites do(s) objecto(s) que se pretende identificar. Um dos pontos relevantes é a inicialização da curva, que em geral poderá ter implicações no resultado final. Em alguns métodos semi-automáticos (ou semi-supervisionados), o operador traça um contorno aproximado que é revisto e melhorado de forma automática. Nesta secção são apresentados de forma muito resumida dois métodos que podem ser implementados de forma automática (sem intervenção de um operador) - um paramétrico (GVF snakes) e um não-paramétrico (Levelsets).

4.1 Contornos Activos / GVF Snakes

O modelo de Contornos Activos conciste na minimização de uma função energia, que depende da forma de uma curva *spline* e da sua localização na imagem. A curva (*snake*) é definida parametricamente como $v(s) = [x(s), y(s)]$, onde $x(s)$ e $y(s)$ são as coordenadas x, y (linha, coluna) ao longo do contorno, e $s \in [0, 1]$. A energia que se pretende minimizar é dada por (7), onde E_{int} é a energia interna da curva e E_{ext} é a energia externa, proveniente de um campo de forças (*Gradient Vector Flow*). A componente interna corresponde a uma tensão que favorece a suavidade da curva, penalizando variações excessivas na sua forma. A componente externa pode ser obtida por exemplo do gradiente da imagem, dando origem a um campo de forças externas que obrigam a *snake* a mover-se para objectos limite (contornos). Normalmente há 1 ou mais parâmetros na função energia (7), que se procura minimizar através de um processo iterativo.

$$E_{snake} = \int_0^1 E_{snake}(v(s)) ds = \int_0^1 \left[E_{int}(v(s)) + E_{ext}(v(s)) \right] ds \quad (7)$$

Na figura 13 é apresentado um exemplo de segmentação de uma imagem de um objecto em forma de U claro, com fundo escuro [Xu]. A imagem original (esquerda) tem bastante ruído, o que dificulta o processo de segmentação. A partir

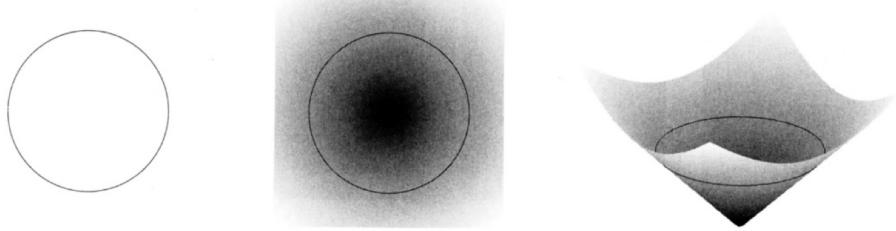


Figura 14: Exemplo de criação de *Level Set* $\Phi(x, y)$ para uma curva de inicialização circular (esquerda). Visualização de $\Phi(x, y)$ como imagem 2D (centro) e versão 3D (direita), sendo a curva correspondente à região $\Phi(x, y) = 0$ [Sonka].

desta imagem foi obtida uma imagem do gradiente, que serviu por base para a criação do campo de forças externas (2^a e 3^a imagens a partir da esquerda). Na figura 13 (direita) é apresentada a *snake* sobreposta à imagem original, para várias fases do processo iterativo, partindo de um contorno inicial que é a curva maior que contém o objecto. Neste exemplo o processo termina com a curva sobreposta ao contorno do objecto em forma de U, como pretendido.

Apesar do processo de segmentação ter sido muito bem sucedido neste caso, há alguns inconvenientes dos métodos de segmentação com *snakes*. Um deles é a necessidade de se ajustar 1 ou mais parâmetros de regularização, e outro é o facto de apenas ser possível obter 1 objecto segmentado. Para além disso, em princípio o contorno inicial deverá conter ou estar contido no objecto pretendido, o que nem sempre é exequível. Ainda assim, em certos casos é possível definir uma região inicial com segurança e definir os parâmetros de regularização com base num pequeno conjunto de imagens de treino.

4.2 Level Sets

Uma outra forma de segmentação de imagem é através de modelos geométricos deformáveis, que se podem chamar contornos geodésicos activos ou propagação por *Level Sets*. A principal diferença relativamente aos modelos de *snakes* é o facto de serem modelos não paramétricos. Uma outra diferença é que do processo de segmentação por *Level Sets* podem resultar múltiplos objectos.

Dado um contorno inicial $v(s, t) = [x(s, t), y(s, t)]$, para $t = 0$, é criada uma função *Level Set* (Φ), cujos valores $\Phi(x, y)$ correspondem à distância (em pixels) ao contorno $v(s, t)$, com atribuição de sinal + no exterior e - no interior. O processo é ilustrado na figura 14, para um contorno circular. A figura mostra o contorno (esquerda) e o *Level Set* $\Phi(x, y)$, representado como uma imagem 2D (centro) e numa visualização 3D (direita). Os níveis de $\Phi(x, y)$ podem ser vistos como elevação ou cota do terreno, com a curva a corresponder a uma cota de 0, i.e. região onde $\Phi(x, y) = 0$.

O processo de segmentação consiste em permitir a evolução de um contorno ao longo do tempo, até que se atinja uma solução estável, sendo os deslocamentos do contorno obtidos a partir do gradiente de Φ . Há várias abordagens para a implementação computacional deste processo iterativo, onde o tempo é convertido para uma variável discreta. O objectivo é obter uma segmentação

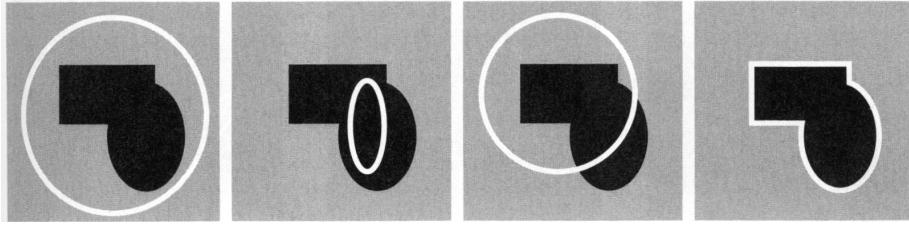


Figura 15: Situações possíveis para o contorno Φ (a partir da esquerda): fora do objecto; dentro do objecto; parte dentro e parte fora; sobre o objecto [Sonka].

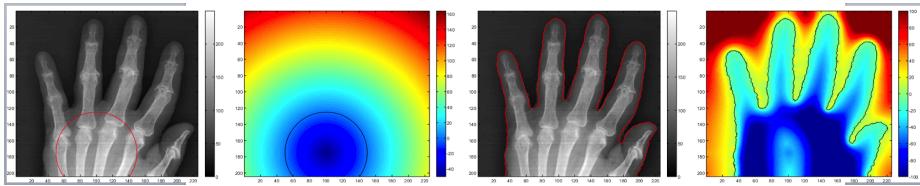


Figura 16: Exemplo de segmentação de imagem com *Level Sets* (a partir da esquerda): imagem original com contorno inicial circular (vermelho); $\Phi(x, y)$ no inicio; contorno final - resultado da segmentação; $\Phi(x, y)$ no final [Svoboda].

satisfatória num intervalo de tempo (número de iterações) razoável. A forma proposta por **Chan & Vase** consiste em minimizar uma função de energia $C(\Phi, a_1, a_2)$, de acordo com (8), onde $I(x, y)$ é a imagem digital e a_1 e a_2 são intensidades médias no interior e exterior do contorno.

$$C(\Phi, a_1, a_2) = \int_{int(\Phi)} \left[I(x, y) - a_1 \right]^2 dx dy + \int_{ext(\Phi)} \left[I(x, y) - a_2 \right]^2 dx dy \quad (8)$$

A energia $C(\Phi, a_1, a_2)$ é minimizada quando o nível 0 de Φ coincide com os contornos da região que melhor separa o objecto do fundo. Considerando os dois termos de C - C_{int} e C_{ext} (8), podem ser identificadas 4 situações, conforme ilustrado na figura 15. Quando a curva está fora do objecto, $C_{int} > 0$ e $C_{ext} \simeq 0$. Quando a curva está dentro do objecto, $C_{int} \simeq 0$ e $C_{ext} > 0$. No caso da curva estar em partes não desprezáveis dentro e fora do objecto, $C_{int} > 0$ e $C_{ext} > 0$. Por último, as 2 componentes C_{int} e C_{ext} tem componentes próximas de 0, minimizando $C(\Phi, a_1, a_2)$, quando a curva está sobre o objecto.

Podem ser usadas outras características locais para a função de energia, para além da intensidade dos pixels, e podem ser considerados parâmetros de regularização em (8), relacionados com o comprimento da curva ou com a área no seu interior.

A figura 16 mostra um exemplo de segmentação de uma imagem com *Level Sets*. Do lado esquerdo é apresentada a imagem original com o contorno inicial (sobreposto a vermelho), e o *Level Set* $\Phi(x, y)$ correspondente. Do lado direito é apresentado o contorno final (resultado da segmentação) sobre a imagem original, e o respectivo *Level Set* $\Phi(x, y)$.

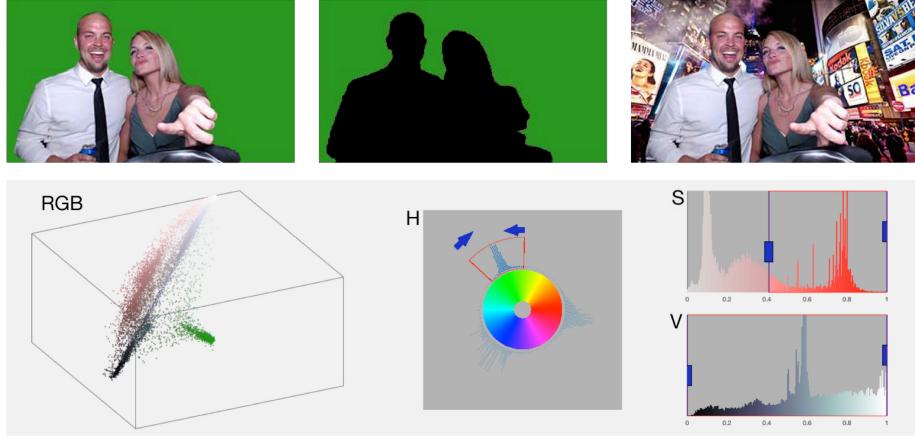


Figura 17: Criação de fotografia com fundo falso por segmentação de imagem de cores. Em cima: imagem original com fundo verde (esquerda); segmentação do fundo (centro); fotografia com fundo novo (direita). Em baixo: localização dos pixels da imagem original no espaço RGB (esquerda) e HSV (direita).

5 Segmentação de Imagens de Cor

Os métodos apresentados nas secções anteriores fornecem alternativas para segmentação de imagens de tons de cinzento (1 banda). No caso de imagens de cor, por exemplo cor RGB com 3 bandas, uma abordagem possível é converter a imagem para uma versão de tons de cinzento antes de aplicar o processo de segmentação. No entanto, esta abordagem é algo limitativa, uma vez que a informação de cor pode ser muito relevante no processo de identificação / descriminação dos objectos de interesse na imagem. Um exemplo onde a cor é particularmente útil é a criação de fotografias ou vídeos com fundos falsos, onde as imagens são adquiridas deliberadamente com o objecto de interesse (normalmente pessoas) em frente de um fundo de cor contrastante, habitualmente verde. Este método é usado em estúdios de televisão, por exemplo para apresentação de meteorologia, ou em muitos locais turísticos, como o Sea Life Centre no Porto.

A figura 17 mostra um exemplo de uma fotografia de 2 pessoas obtida com um fundo verde (em cima à esquerda), com o objectivo de se criar uma versão da fotografia onde as pessoas aparecem com um fundo falso (em cima à direita). Como se pode ver na representação dos pixels da imagem original no espaço RGB (figura 17, em baixo à esquerda), há uma zona de grande densidade de ocupação correspondendo ao fundo (pixels verdes). O objectivo do processo de segmentação é dividir a imagem original em 2 regiões, ou segmentos: um correspondendo aos pixels de fundo, e outros aos pixels do objecto (as pessoas).

Uma forma simples de segmentar imagens de cor é estendendo o método de *Thresholding Global* (2) a imagens de 3 bandas. A aplicação de múltiplas operações de *Thresholding* dá origem a várias imagens binárias, que são depois combinadas através da interseção (ou multiplicação). No exemplo da figura 17 foram usados 3 *Thresholds* no modelo de cor HSV: $T_{H1} = 0.246$, $T_{H2} = 0.382$, e $T_S = 0.408$. Na imagem binária final, que corresponde ao fundo verde, os pixels

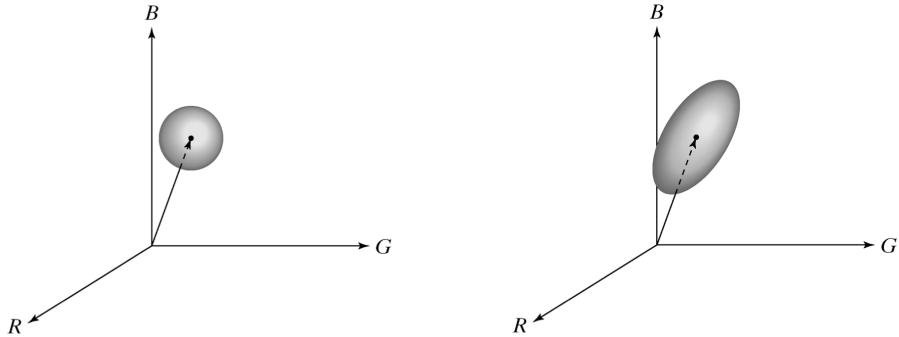


Figura 18: Região associada a um critério de distância a um ponto de cor RGB, para a distância Euclideana (esquerda) e Mahalanobis (direita).

que tomam valor 1 (ON) satisfazem as condições $T_{H1} < H < T_{H2}$ e $S > T_S$. A figura 17 (em cima ao centro) mostra o produto desta imagem binária pela imagem original, onde aparecem inalterados apenas os pixels de fundo, estando os restantes pixels a preto (0).

Uma outra abordagem para segmentação de imagens de cor consiste em usar uma métrica (distância) num espaço de cor, tipicamente no espaço RGB. Seja \mathbf{m} um vector com as componentes R,G,B médias de uma zona de referência. No exemplo da figura 17, \mathbf{m} seria a média dos valores R,G,B de uma amostra de fundo verde. Pode calcular-se a distância Euclideana (D_E) através de (9), onde \mathbf{z} é o vector RGB associado a um pixel observado. O critério de segmentação poderá ser $D_E < T$, onde T (*Threshold*) corresponde ao raio da esfera no espaço RGB associada à região de interesse.

$$D_E = \|\mathbf{z} - \mathbf{m}\| = \left[(\mathbf{z} - \mathbf{m})^T (\mathbf{z} - \mathbf{m}) \right]^{1/2} \quad (9)$$

Uma outra métrica que em princípio poderá ser mais eficaz é a distância Mahalanobis (D_M), calculada através de (10), onde \mathbf{C} é a matriz de covariância dos dados de referência (pixels da imagem RGB pertencentes à zona de referência). A figura 18 mostra o que poderá ser a região correspondente aos valores RGB considerados para a região segmentada, usando como métricas a distância Euclideana (esquerda) e a distância Mahalanobis (direita).

$$D_M = \left[(\mathbf{z} - \mathbf{m})^T \mathbf{C}^{-1} (\mathbf{z} - \mathbf{m}) \right]^{1/2} \quad (10)$$

Há várias formas de usar informação relativa a cor nos diferentes métodos de segmentação. Por exemplo, nos métodos de crescimento de regiões (3.1) e divisão de regiões (3.2), a propriedade \mathcal{P} pode ter por base uma ou mais características de cor. Uma forma poderá ser usando as distâncias D_E ou D_M , com a imposição de que pixels da mesma região terão de ter diferenças de cor inferiores a um certo valor limite.

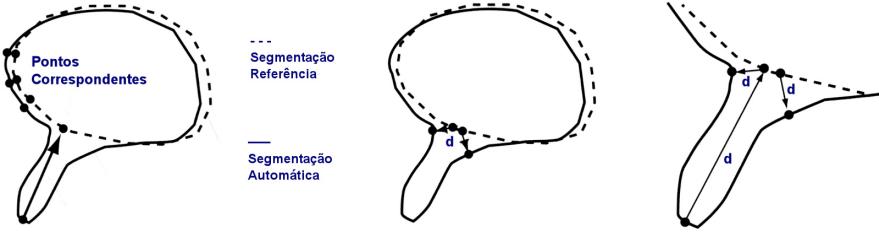


Figura 19: Esquema da abordagem de avaliação de segmentação por contornos.

6 Avaliação de segmentação

Um aspecto particularmente importante é a avaliação da qualidade da segmentação. Esta tarefa pode ser feita não apenas para verificar se o resultado obtido é satisfatório, dentro de uma margem de erro considerada aceitável, mas também para selecionar o método mais adequado para um conjunto de dados, ou para ajustar (treinar) 1+ parâmetros que não estejam definidos a priori. Há no entanto algumas questões potencialmente difíceis relacionadas com a avaliação de segmentação que tem de ser consideradas:

1. Qual é a segmentação ideal?
2. Como quantificar a diferença (erro) entre 2 segmentações?

Em relação ao ponto 1, não é normalmente fácil ter uma segmentação de referência, e mesmo que exista poderá não ser a ‘ideal’, o que limita a qualidade da avaliação com base nessa referência. Em termos de quantificação da diferença entre um segmentação a avaliar (S_A) e uma de referência (S_R), há essencialmente 2 tipos de abordagens distintas: por contornos e por áreas.

6.1 Avaliação por Contorno

Na avaliação por contornos pretende-se que o desvio entre os contornos de S_A (conjunto de pontos A) e S_R (conjunto de pontos R) seja o menor possível. A abordagem é ilustrada na figura 19, onde a segmentação a avaliar (automática) é apresentada em linha cheia e a referência a tracejado.

Uma forma de quantificar a diferença entre as 2 segmentações é através da distância de Hausdorff (D_H), definida por (11, 12), onde $d(a, r)$ é a uma medida de distância entre os pontos a (de A) e r (de R).

$$h(A, R) = \max_{a \in A} (\min_{r \in R} d(a, r)) \quad (11)$$

$$D_H(A, R) = \max(h(A, R), h(R, A)) \quad (12)$$

O valor de D_H corresponde à maior distância entre os contornos de S_A e S_R , desde que estes estejam definidos por um elevado número de pontos, praticamente adjacentes considerando a resolução da imagem digital. No exemplo da figura 19, os contornos são muito próximos com exceção de uma pequena zona, que acaba por ser a que contribui para o valor da distância de Hausdorff.



Figura 20: Comparação de segmentação por áreas - representação esquematica das 4 situações possíveis (esquerda), exemplo de segmentação de referência e automática (direita).

6.2 Avaliação por Área

A avaliação de segmentação por áreas é bastante mais simples de implementar computacionalmente do que a avaliação por contornos. Cada segmentação (S_A e S_R) é representada por uma imagem binária, e para cada pixel p , há 4 situações possíveis, como ilustrado no esquema da figura 21 (esquerda):

- $p \in S_A$ e $p \in S_R$, verdadeiros positivos (*True Positive* - TP)
- $p \in S_A$ e $p \notin S_R$, falsos positivos (*False Positive* - FP)
- $p \notin S_A$ e $p \in S_R$, falsos negativos (*False Negative* - FN)
- $p \notin S_A$ e $p \notin S_R$, verdadeiros negativos (*True Negative* - TN)

Com base nas imagens binárias, é fácil calcular o número de pixels em cada uma das 4 situações. Idealmente pretende-se ter TP (e também TN) alto, e FP e FN o mais baixo possível. No entanto, dependendo da situação poderá ser mais penalizador ter FP ou FN. Há vários índices de avaliação de segmentação baseados nestas 4 medidas, tais como: True detection rate (TDR), True negative rate (TNR), False positive rate (FPR), False negative rate (FNR). Destes, os mais usados são o TDR (13), caso se pretenda evitar falsos negativos (sensibilidade alta), e o TNR (14), caso se pretenda evitar falsos positivos (especificidade alta). Estes índices tomam valores entre 0 e 1, com uma segmentação perfeita a corresponder ao valor 1.

$$TDR = \frac{TP}{TP \cup FN} = \frac{S_A \cap S_R}{S_R} \quad (13)$$

$$TNR = \frac{TN}{FP \cup TN} = \frac{\overline{S}_A \cap S_R}{S_R} \quad (14)$$

Um outro índice que toma valores entre 0 e 1 (valor 1 para o caso ideal), mas que penaliza igualmente falsos positivos e falsos negativos, é o índice de semelhança de Jaccard (J), calculado por (15). Este é um índice particularmente interessante caso se considere igualmente mau ter sub- ou sobre- segmentação.

$$J = \frac{TP}{FP \cup TP \cup FN} = \frac{S_A \cap S_R}{S_A \cup S_R} \quad (15)$$

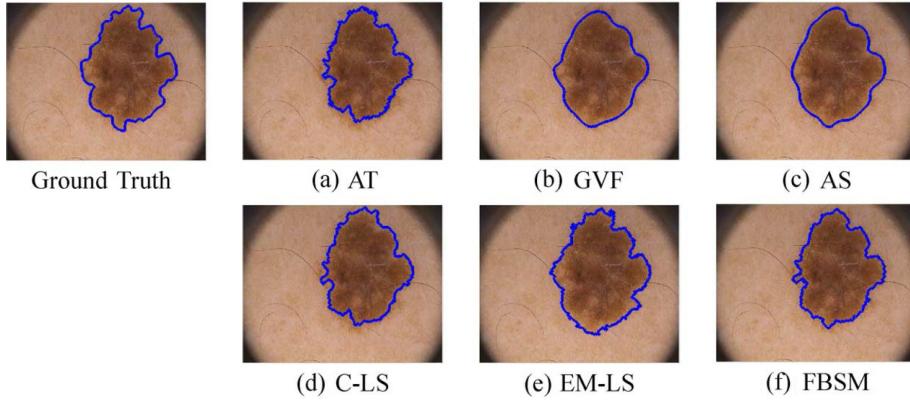


Figura 21: Comparação de segmentação de imagem de dermoscopia, por áreas. Segmentação de referência (em cima à esquerda); segmentação automática obtida por 6 métodos diferentes (a-f) [Silveira].

O índice de Hammoude (H), calculado por (16), também penaliza igualmente os dois tipos de falhas na segmentação. Este índice toma valores entre 0 e 1, mas uma segmentação perfeita corresponde neste caso ao valor 0. Na verdade o índice de Hammoude é o índice de Jaccard com a escala de valores invertida, isto é $H = 1 - J$.

$$H = \frac{FP \cup FN}{FP \cup TP \cup FN} = \frac{(S_A \cup S_R) \setminus (S_A \cap S_R)}{S_A \cup S_R} \quad (16)$$

A figura 21 mostra um exemplo de comparação de resultados de segmentação de uma imagem de dermoscopia, por áreas, onde é apresentada a segmentação de referência (em cima à esquerda), definida manualmente por um médico dermatologista, e segmentações automáticas obtida pelos seguintes métodos [Silveira]:

- (a) Adaptive Thresholding (AT)
- (b) Gradient Vector Flow (GVF)
- (c) Adaptive Snake (AS)
- (d) Level set method of Chan et al. (C-LS)
- (e) Expectation-Maximization Level Set (EM-LS)
- (f) Fuzzy-Based Split-and-Merge algorithm (FBSM)

Como se pode observar na figura 21, a região segmentada é diferente para cada método, havendo em todos os casos alguma diferença em relação à segmentação de referência.

7 Segmentação no MATLAB

Nesta secção são apresentadas algumas funções MATLAB relacionadas com segmentação de imagem.

A aplicação de *Thresholding Global* (2) pode ser feita com a função `imbinarize`, sendo necessário atribuir um valor para o *threshold* (T). A função `graythresh` permite calcular o valor de T através do método de Otsu (2.1.1). Em seguida um exemplo para a imagem de níveis de cinzento IM.

```
>> T0tsu = graythresh(IM);  
>> IB = imbinarize(IM,T0tsu);
```

calcula valor
de T Otsu

imagem
binária

A segmentação por contornos activos (7) pode ser feita recorrendo à função `activecontour`. É necessário indicar a imagem a segmentar e a região inicial (`mask`), sendo o resultado uma imagem binária (`IB`). Em seguida um exemplo de execução para uma imagem de níveis de cinzento IM.

```
>> IB = activecontour(IM,mask)
```

7.1 Segmentação com colorThresholder

O MATLAB tem uma ferramenta interactiva muito útil para a segmentação / binarização de imagens a cores – `colorThresholder`. A aplicação permite a utilização dos modelos de cor RGB, HSV, YCbCr e L*a*b*. O operador começa por escolher um modelo de cor, e em seguida pode experimentar valores limite (*thresholds*) para as várias componentes de cor, e observar o resultado da binarização com essas restrições. No final pode exportar a imagem binarizada ou criar uma função que aplica os *thresholds* escolhidos a uma imagem de *input*.

7.2 Aplicação imageSegmenter

A aplicação `imageSegmenter` tem uma série de funcionalidades para segmentação manual e semi-automática de imagens. Permite utilizar métodos de segmentação e refinar os resultados, ou fazer uma segmentação manual assistida por ferramentas de apoio à preparação de dados vectoriais. No final permite exportar a segmentação criada.

8 Referências Bibliográficas

- Beucher** Serge Beucher, The Watershed Transformation, 2010, Centre for Mathematical Morphology the image processing laboratory of MINES ParisTech.
<http://www.cmm.mines-paristech.fr/beucher/wtshed.html>
- Chan & Vase** Chan, T.F., Vase, L.A., 1997, Active contour without edges, IEEE Transactions on Image Processing, Vol.10(2), 266-277.
- Gonzalez** Digital image processing, R.C. Gonzalez, R.E. Woods, 3rd Ed., Prentice Hall (2008).
- GonzalezM** Digital image processing using MATLAB, R.C. Gonzalez, R.E. Woods, S.L. Eddins, 2nd Ed., Gatesmark Publishing (2009).
- MathWorks** The MathWorks, MATLAB and Image Processing Toolbox, Release 2019a, Natick, Massachusetts, United States, MathWorks (2019).
- Otsu** Otsu, N., 1979, A Threshold Selection Method from Gray-Level Histograms, IEEE Transactions on Systems, Man, and Cybernetics, Vol.9(1), 62-66.
- Silveira** Silveira, M., Nascimento, J.C., Marques, J.S., Marçal, A.R.S, Mendonça, T., Yamauchi, S., Maeda, J., Rozeira, J., 2009, Comparison of Segmentation Methods for Melanoma Diagnosis in Dermoscopy Images, IEEE Journal of Selected Topics in Signal Processing, Vol.3(1), 35-45.
- Sonka** Image Processing, Analysis, and Machine Vision, M. Sonka, V. Hlavac, R. Boyle, 3rd ed., Cengage Learning (2008).
- Svoboda** Image Processing, Analysis, and Machine Vision - A MATLAB Companion, T. Svoboda, J. Kybic, V. Hlavac, Thomson (2008).
- Xu** Xu C., Prince J.L., 1998, Snakes, Shapes and Gradient Vector Flow, IEEE Transactions on Image Processing, Vol.7, 359-369.

9 Exercícios propostos

Nesta secção são propostos alguns exercícios para MATLAB que poderão ser usados para consolidar os conceitos de Processamento de Imagem apresentados.

I - *Thresholding Global*

Crie uma função MATLAB que receba como *input* o nome de um ficheiro imagem (formato *standard*), importe a imagem, e no caso de ser RGB a converta para imagem de tons de cinzento, com 1 banda (IC). A partir de IC, crie 2 imagens binárias por *Thresholding Global*: uma com T_g obtido pelo método de Otsu (IBO) e outra usando o valor da mediana de IC para T_g (IBM). Deverá ser apresentada uma figura com as seguintes elementos (num **subplot** de 2×2):

- imagem de tons de cinzento (IC)
- histograma de IC
- imagem binária obtida com $T_g = T_{Otsu}$ (IBO)
- imagem binária obtida com $T_g = T_{Mediana}$ (IBM)

Deverão ser apresentados os valores de T_{Otsu} e $T_{Mediana}$, no domínio uint8 (0-255), no título das imagens binárias.

Exemplo de utilização: `TGlobal('Urso.jpg')`

II - *Thresholding Local*

Crie uma função MATLAB para fazer *Thresholding Local*, dividindo a imagem em $S \times S$ sub-imagens, de tamanho aproximadamente igual. A função deverá receber os seguintes inputs: (1) o nome de um ficheiro imagem (formato *standard*); (2) o valor para S (inteiro de 1 a 20); (3) parâmetro GRAF que controla a apresentação de resultados. No caso da imagem original ser RGB, deverá ser convertida para imagem de tons de cinzento, com 1 banda (IC). O output da função é a imagem binária obtida por *Thresholding* de IC. Se GRAF=1, deverá ser mostrado numa janela a imagem IC, e as binárias obtidas por *Thresholding Global* (Otsu) e Local (num **subplot** de 1×3).

Sintaxe: `TLocal(Ficheiro,S,GRAF)`

Exemplo de utilização: `TLocal('Urso.jpg',3,1)`

III - Avaliação do desempenho de `TLocal`

Escreva um script que teste o desempenho do algoritmo de *Thresholding local*, usando a função `TLocal`, simulando por exemplo diferentes condições de iluminação e considerando vários valores de S .

9.1 APPENDIX - Exercices (in English)

This Appendix proposes some exercises for MATLAB that can be used to consolidate the concepts of Image Segmentation.

I - Global Tresholding

Create a MATLAB function that receives as input an image file name (standard format), imports the image, and if it is RGB converts it to a grayscale image, 1 band (IC). From IC, create 2 binary images by Global Tresholding: one with T_g obtained by the Otsu method (IBO) and another using the median value of IC for T_g (IBM). A window should be displayed with the following elements (2×2 subplot):

- grayscale image (IC)
- histogram of IC
- binary image obtained with $T_g = T_{Otsu}$ (IBO)
- binary image obtained with $T_g = T_{Median}$ (IBM)

The values of T_{Otsu} and T_{Median} should be presented in the titles of the binary images, in the uint8 domain (0-255).

Example of use: `TGlobal('Urso.jpg')`

II - Local Tresholding

Create a MATLAB function to perform Local Tresholding, dividing the image into $S \times S$ sub-images, of approximately equal size. The function should receive the following inputs: (1) the image file name (*standard* format); (2) the value for S (integer from 1 to 20); (3) GRAF parameter that controls the presentation of results. If the original image is RGB, it must be converted to a grayscale image, with 1 band (IC). The function output is the binary image obtained by Tresholding of IC. If GRAF=1, a window should be presented with the image IC, and the binary images obtained by Global Tresholding (Otsu) and Local Tresholding (1×3 subplot).

Syntax: `TLocal(FileName, S, GRAF)`

Example of use: `TLocal('Urso.jpg', 3, 1)`

III - Evaluation of the TLocal performance

Write a script that tests the performance of the Local Tresholding algorithm, using the function `TLocal`, simulating for example different lighting conditions and considering various values for S .