

**Universidade do Minho**

Escola de Engenharia

|  |  |  |
| --- | --- | --- |
| Diogo Nogueira A78957 | Gonçalo Costa PG42839 | Guilherme Martins A70782 |

***Detection and Location Foot***

*Second Tutorial* *of* *Vision Computer*

**Mestrado (Integrado) em Eng. Informática** 2020/2021

***Tutorial submitted by:***

**Conteúdo**

[*Detection and Location Foot* 3](#_Toc62142302)

[Algoritmo 3](#_Toc62142303)

[Análise dos Resultados 9](#_Toc62142304)

[Resultados Obtidos 9](#_Toc62142305)

[Comparação com os Resultados *Ground Truth* 10](#_Toc62142306)

# *Detection and Location Foot*

A ideia deste Tutorial passa por desenvolver um algoritmo que seja capaz de detetar a zona dos pés, particularmente as articulações correspondentes ao **tornozelo**e à **ponta do pé**, através do uso de métodos e ferramentas de processamento de imagem clássico.

## Algoritmo

Antes de inicializar todas as fases intrínsecas do algoritmo criado, teve de existir uma escolha acerca do tipo de ***input*** a utilizar e, assim, entender de que maneira esse ***input*** poderia ou não influenciar na obtenção dos resultados pretendidos.

Pela análise das hipóteses fornecidas no enunciado deste Tutorial 2, existiu a decisão de se usar a Imagem **“gait\_depth\_oneimage.png”**, correspondente ao Mapa de Profundidade da Imagem Original, uma vez que isso acabaria por facilitar o processamento posterior da Imagem.

|  |  |
| --- | --- |
| Uma imagem com interior  Descrição gerada automaticamente | Uma imagem com escuro, aceso, noite, céu noturno  Descrição gerada automaticamente |
| **Figura 1** Imagem Original (RGB) | **Figura 2** Imagem Depth |

**Este algoritmo exigiu a criação de dois *scripts* aliciantes – o *script* principal “detectFoots.m” e o *script* secundário “coordRectangle.m”, cuja utilidade será detalhada no estender da explicação do algoritmo.**

1. **Leitura da Imagem *Depth***

**Ajuste do Contraste da Imagem *Depth***

**Corte da Imagem *Depth***

Para facilitar o processamento da Imagem *Depth*, fez-se um corte aleatório e incidente na zona das pernas/pés. Este corte é feito de modo centralizado, viabilizando a recuperação do tamanho Original (e respetivas coordenadas) da Imagem usada como *input*.

O corte é processado após a obtenção da Imagem *Depth* ajustada em termos de contraste, pelo uso da função MATLAB **imadjust()**. Esta função ajusta o contraste da Imagem, de forma a que 1% dos seus dados se tornem saturados em baixas e altas intensidades.

**Fica assim criado/desenvolvido o método basilar do algoritmo, já que apronta a Imagem que será usada ao longo do *script*, tornando-a perfeitamente observável em termos de dados a constituir.**

|  |  |
| --- | --- |
|  |  |
| **Figura 3** Imagem Depth com ajuste Contraste | **Figura 4** Imagem Depth Cropped |

1. **Transposição da Matriz da Imagem *Cropped***

**Valor Pixel Máximo por cada Linha da Imagem**

Uma vez que se sabe que a Imagem em MATLAB pode (e deve) ser interpretada em modo Matriz/Pixéis, é necessário criar a transposição da Matriz relativa à Imagem *Cropped*.

Note-se que apenas com a transposição desta Matriz é possível recorrer à função MATLAB **max()**, uma vez que esta função obtém o valor máximo por **coluna**e não por linha.

1. **Binarização da Imagem *Cropped* por Cada Linha**

Por recurso a um processo iterativo, a técnica de Binarização permitirá obter uma Imagem totalmente a Preto e Branco, onde estarão preenchidos a Branco apenas os Pixéis pertencentes à zona das Pernas/Pés.

**Assim, e uma vez que existe já uma Matriz contendo o Valor Máximo de Cada Linha da Imagem *Cropped* (obtida nos passos anteriores), efetua-se a seguinte verificação (tendo em conta uma Margem de Erro):**

matrixBinary(row, column) <= (maxForLineCroppedImage(1, row) - errorMargin)

Todos os Pixéis que satisfaçam esta condição serão preenchidos a Branco, deixando a Preto os restantes. Este cálculo é considerado através da Ideia de Profundidade da Imagem e, permite, de uma forma muito eficiente, destacar apenas a Área de Interesse para estes primeiros passos.

|  |  |
| --- | --- |
|  |  |
| **Figura 5** Imagem Depth Cropped | **Figura 6** Imagem Depth Cropped Binarize |

1. **Aplicação de Efeitos de Erosão na Imagem *Cropped***

**Obtenção Coordenadas X e Y dos Pixéis a Branco**

**Separação das Pernas e Binarização da Imagem de Cada Perna**

A aplicação de um Efeito de Erosão sobre a Imagem surge com a necessidade de individualizar ambas as pernas, para num passo seguinte ocorrer uma limitação das *boundaries* de cada Pé.

Isso implica também obter as Coordenadas X e Y dos Pixéis a Branco da Imagem *Depth Cropped Binarize*, que permitirão usar a função MATLAB **bwselect()** e com isso selecionar as duas partes da Imagem de forma individual.

Com as duas metades da Imagem, entra-se num novo processo de Binarização e obtém-se, finalmente, uma nova Área de Interesse, que corresponde agora aos Pés.

**Considera-se assim concluído o Processamento da Imagem e entra-se agora no Processo de Deteção de Coordenadas em si.**

|  |  |
| --- | --- |
|  |  |
| **Figura 7** Imagem Depth Cropped Binarize | **Figura 8** Imagem Depth Cropped Binarize com Efeito de Erosão |

**Repare-se que a Imagem com o Efeito de Erosão aplicado está menos condensada em termos de Pixéis Brancos. Isso é de extrema importância, já que só dessa maneira é possível criar uma separação entre as duas pernas.**

**Caso contrário, ao usar-se a função bwselect() o resultado seria exatamente o mesmo daquele que se observa na Figura 7.**

1. **Coordenadas das *Boundaries* de Cada Pé**

É neste passo que entra o *script* **“coordRectangle.m”**, cujo propósito passa por determinar as Coordenadas X e Y que depois de interligadas conseguem desenhar um retângulo à volta de cada Pé.

O algoritmo imposto nesse *script* é um algoritmo totalmente adaptado e refletido para funcionar neste projeto, cuja base foi obtida no Fórum Oficial MATLAB, mas que permite apenas obter o que é importante para ingressar na Fase Final descrita na alínea que se segue.

1. **Interpretação das Coordenadas Finais**

**Desenho na Imagem Original**

|  |  |
| --- | --- |
| **P1**  **P2**  **P3**  **P4**  **P1** (x1, y1)  **P2** (x2, y2)  **P3** (x3, y3)  **P4** (x4, y4) | **P4**  **P1**  **P2**  **P3** |
| **Figura 9**Interpretação Coordenadas *Boundaries* do Pé Esquerdo | **Figura 10** Interpretação Coordenadas *Boundaries* doPé Direito |

|  |  |
| --- | --- |
|  |  |
| **Figura 11** Pé Esquerdo com as Boundaries desenhadas | **Figura 12** Pé Direito com as Boundaries desenhadas |

A análise das Figuras anteriores é crucial para se conseguir, finalmente, calcular as coordenadas que dizem respeito às pontas e tornozelos de ambos os Pés.

**Depressa se nota que o “fluxo” de coordenadas e a forma como são devolvidas pelo *script*/função “coordRectangle.m” é diferente para cada um dos pés, valores estes influenciados pela posição em que cada um dos mesmos se encontra disposto – o Pé direito apresenta uma ligeira rotação no sentido dos ponteiros do relógio.**

**480**

|  |
| --- |
| **Uma imagem com exterior, rebanho  Descrição gerada automaticamenteUma imagem com interior  Descrição gerada automaticamente**  **150**  **300** |

**6409**

Com a observação dos valores das dimensões de cada uma das imagens, torna-se possível definir a margem entre a **largura** e **comprimento** existente entre as mesmas e, dessa forma, acrescentar esses valores às coordenadas que correspondem às pontas e tornozelos.

**Para obter essas coordenadas, assumiu-se, por estimativa, que tanto a ponta do pé como o tornozelo estariam localizados exatamente ao meio da linha da *boundary* de cada extremidade (conforme sinalizado pelas bolas a vermelho na Figura 11 e 12).**

## Análise dos Resultados

### Resultados Obtidos

Estando as coordenadas totalmente recuperadas e adaptadas à Imagem Original, foi apenas necessário desenhá-las sobre a mesma. Depois, recorrendo a funções basilares do MATLAB, introduzir esses mesmos valores num ficheiro Excel, para poder facilitar no processo de comparação de resultados.



**Figura 13** Imagem Original com Desenho das Coordenadas das Pontas do Pé e Tornozelos

### Comparação com os Resultados *Ground Truth*

|  |  |  |  |
| --- | --- | --- | --- |
|  | Valores Obtidos | Valores *Ground Truth* | Diferença |
| Tornozelo Esquerdo | | | |
| X | 307 | 266 | 41 |
| Y | 197 | 350 | 153 |
| Ponta Pé Esquerda | | | |
| X | 279 | 256 | 23 |
| Y | 249 | 302 | 53 |
| Tornozelo Direito | | | |
| X | 360 | 324 | 36 |
| Y | 135 | 269 | 134 |
| Ponta Pé Direta | | | |
| X | 362 | 338 | 24 |
| Y | 180 | 232 | 52 |



**Figura 14** Sobreposição Imagem com os Resultados Obtidos (Vermelho), Resultados de Referência (Amarelo) e Resultados Ground Truth Excel (Rosa)

**A comparação de resultados foi pensada para conseguir albergar todo o leque de dados disponível e concluir, com isso, o sucesso deste algoritmo de deteção e localização dos pés.**

* Os Resultados da Tabela comparativa demonstram uma diferença ainda substancial;
* Pela análise dos Resultados Obtidos (Vermelho) em comparação com os Resultados da Imagem de Referência (Amarelo), percebemos que a diferença de valores da Tabela possa estar a ser influenciada por algum tipo de “desvio” nos valores;
  + O desenho dos Resultados de *Ground Truth* (Rosa) valida esse desvio, já que demostram que esses valores não correspondem exatamente à ponta e tornozelo de cada um dos pés.
* **Assim se conclui que os Resultados Obtidos são uma excelente aproximação em termos das Coordenadas pedidas, sendo normal existir uma diferença mínima em relação aos valores 100% exatos.**