Visão por Computador, 2016-2017
Trabalho Prático Final - Classificação de folhas

Rui Oliveira, Tomás Rodrigues
DETI, Universidade de Aveiro
Aveiro, Portugal
{ruipedrooliveira, tomasrodrigues}@ua.pt

Resumo –

Pretende-se através deste relatório expor sob forma escrita, o nosso desempenho e objetivos alcançados no trabalho prático final da unidade curricular de Visão por Computador do Mestrado Integrado de Engenharia de Computadores e Telemática. Neste relatório pretenderemos explicar algumas das opções por nós encontradas de modo a dar resposta ao problema que iremos descrever de seguida.

***Palavras chave* – visão, computador, imagem digital, classificação, svm, características, python, opencv**

I. ENQUADRAMENTO

O trabalho que o nosso grupo desenvolveu no âmbito da unidade curricular ”Visão por Computador”está enquadrado no projeto ”EduPARK - Mobile Learning, Realidade Aumentada e Geocaching na Educação em Ciências - um projeto inovador de investigação e desenvolvimento”financiado por fundos do COMPETE 2020. O desafio do EduPArk consiste em criar estratégias originais, atrativas e eficazes de aprendizagem interdisciplinar em Ciências. Estes objectivos serão alcançados através da criação de uma aplicação interativa em Realidade Aumentada, com recurso a dispositivos móveis, suportando atividades baseadas em Geocaching. A aplicação será explorada por professores e alunos desde o ensino básico ao superior, em contextos de atividades ao ar livre, com potencial utilidade também no domínio do turismo/público em geral.

O nosso trabalho consistiu num estudo das características a ter em consideração na identificação de uma folha de árvore de uma determinada espécie. Após a recolha e organização das características encontradas, usámos um classificador que permitiu identificar a espécie a que pertence uma determinada folha.

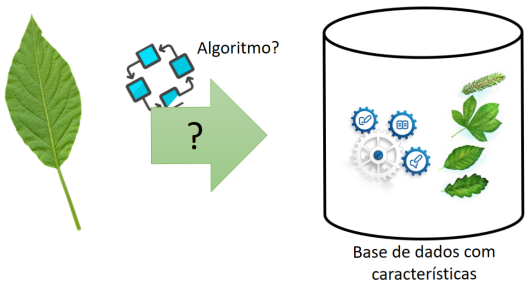

Figura 1: Problema geral

II. FUNDAMENTOS TEÓRICOS

Antes de procedermos à implementação deste projeto foi necessário consultar alguma literatura de modo a definir quais as características essências a recolher para a identificação de uma folha numa imagem digital. Para tal, consultámos alguns projetos *opensource* existentes e alguns artigos/sites que tentam discutir algumas dessas características.

A. Características extraídas de uma folha

As características que achámos relevantes no contexto do nosso projeto serão descritas de seguida. Todas estas caraterísticas são extraídas de imagens com um pré-processamento que iremos explicar mais à frente neste relatório.

A.1 Número de cantos

Os cantos em processamento de imagem é umas das características mais importantes. Consiste na intersecção de duas arestas ou seja representa um ponto em que as direções dessas duas arestas muda. Neste contexto será utilizádo o método `cv2.goodFeaturesToTrack()` em *Shi-Tomasi Corner Detector Good Features to Track* disponivel em `http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_shi_tomasi/py_shi_tomasi.html`

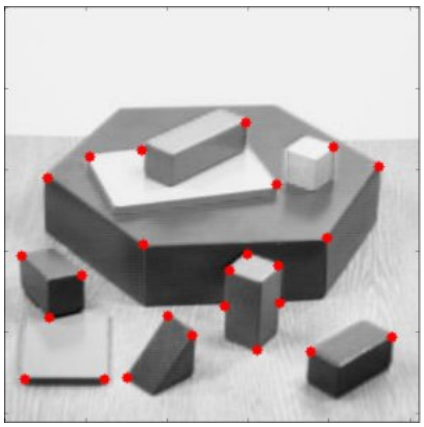

Figura 2: Utilização do método goodFeaturesToTrack()

A.2 Razão entre comprimento e altura

Consiste na relação entre a largura e a altura da delimitação do objeto ou seja a relação proporcional entre a largura e a altura. É calculada através da seguinte expressão:

$$AspectRatio = \frac{Width}{Height}$$

Exemplo no tutorial[1] do openCV.

```
x,y,w,h = cv2.boundingRect(cnt)
aspect_ratio = float(w)/h
```

A.3 Razão entre perímetro e área

De forma análoga à característica anterior, esta permite-nos verificar a relação de proporcionalidade existente entre o perímetro e a área da folha.

$$rpa = \frac{perimetro}{area}$$

rpa: razão entre perímetro e área

A.4 Razão entre áreas

Consiste na relação entre as áreas internas e externas ao contorno da folha.

$$ra = \frac{area1}{area2}$$

ra: razão entre áreas

A.5 Solidez

A solidez é a razão entre a área do contorno e a sua área convexa formada pela união dos pontos do polígono calculado. Por exemplo na imagem seguinte será á área da mão sobre a área do polígono desenhado a vermelho.

$$Solidity = \frac{ContourArea}{convexHullArea}$$

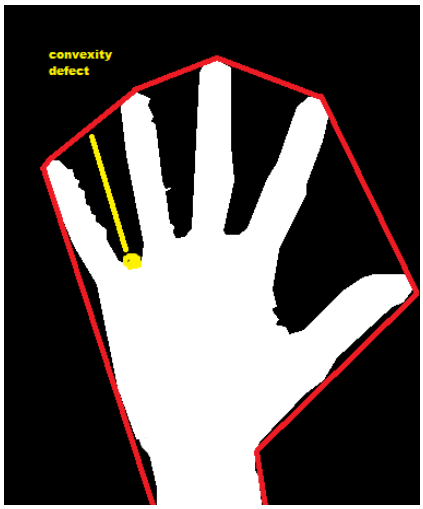

Figura 3: Exemplo cálculo da solidez

[1]http://docs.opencv.org/trunk/d1/d32/tutorial_py_contour_properties.html

A.6 Momentos

Os momentos em imagens ajudam-nos a calcular algumas características como o centro de massa do objeto, áreas do objeto (ou intensidade total) e informações sobre a sua orientação. Define-se como sendo uma média ponderada especial (momentos matemáticos[2]) das intensidades dos pixeis de uma imagem ou uma função esses momentos. Os momentos são úteis para descrever objetos depois da segmentação.

Figura 4: Imagem input, exemplo. Tutorial OpenCV

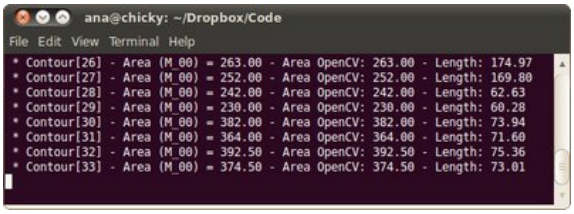

Figura 5: Resultado, exemplo. Tutorial OpenCV

III. PROCEDIMENTOS E ESTRUTURA

Este projeto têm como principal objetivo a resolução do dois seguintes problemas:

- Extração das características de uma imagem
- Utilização de um algoritmo de classificação que permita identificar uma folha com base nas características extraídas do ponto anterior

O ficheiro `training_Leaves.py` permite extrair as características de um conjunto de imagens localizados nas pastas `Data/DataTraining/<nomefolha>`.

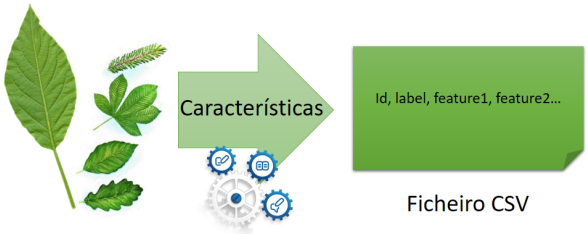

Figura 6: Exemplo de resultado final

Essas características são guardadas num ficheiro do tipo CSV no diretório `Data/Training_Tables` com as seguintes entradas: id, label, caracteristicas...

[2]https://en.wikipedia.org/wiki/Moment_(mathematics)

	id	label	number_corners	length_width_ratio	perimeter_area_ratio	ratio_of_areas	solidity	mu02	mu03
1	1	Acer_platanoides	22	1.60103626943	0.403494207766	55.1340138108	0.523424099838	5332496955O.7	6.64652797279e+12
2	2	Acer_platanoides	53	1.63001240182	0.41141949143	86.1167319742	0.447832445344	611007314359.0	3.16313368848e+14
3	3	Acer_platanoides	27	1.28092783505	0.409558004209	54.570756469	0.522850288505	47910792572.0	1.00351232489e+12
4	4	Acer_platanoides	29	1.61773255814	0.402451920492	54.8989299733	0.496183423517	45808535982.0	6.13111977024e+12
5	5	Acer_platanoides	33	1.63769690386	0.405251007552	72.3952341824	0.556717157664	208447567256.0	2.56630831266e+13

Figura 7: Ficheiro CSV exemplo

Após a extração de todas as características de um conjunto alargado de dados é então possível testar algumas imagens. Para tal, é necessário executar o ficheiro `test_Leaves.py` o qual acede às imagens disponíveis no diretório `Test_Leaves` e aos ficheiros CSV anteriormente gerados. Este módulo python acede a um algoritmo de classificação e que nos permite observar a previsão deste.

IV. IMPLEMENTAÇÃO

A. *Tecnologias utilizadas e dependências*

- Linguagem: python 2.7.6
- Processamento de imagem: OpenCV 3.0
- Interface gráfica: PyQt4
- Machine Learning em Python: sklearn

B. *Segmentação e processamento inicial*

Antes de proceder à extração das caraterísticas de uma imagem é realizada uma segmentação com o intuito de minimizar o ruído de uma determinada imagem. Após este processo será retornada uma imagem que contém apenas o contorno da folha.

C. *Aplicar threshold*

Antes de aplicar um determinado threshold é necessário garantir que uma determinada imagem está em grayscale.

Figura 8: Imagem em escala de cinzento

Seguidamente é aplicado um threshold binário invertido, de modo a ficar com a folha a branco e o background todo a preto.

```
ret,thresh = cv2.threshold(imgray,127,
255,cv2.THRESH_BINARY_INV)
```

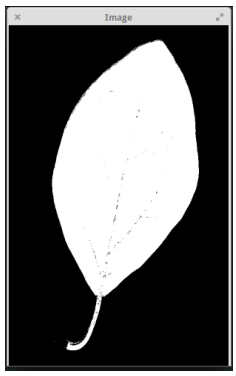

Figura 9: Aplicação de um threshold binário invertido

D. *Aplicar operação morfológica*

Após o threshold aplicámos uma operação morfologia, neste caso um close às imagens a processar de modo a eliminar possíveis ruídos que ainda tenha permanecido, como por exemplo algumas nervura das folhas como se pode observar na figura seguinte.

```
se = ones((15,15), dtype='uint8')
image_close = cv2.morphologyEx(thresh,
cv2.MORPH_CLOSE, se)
```

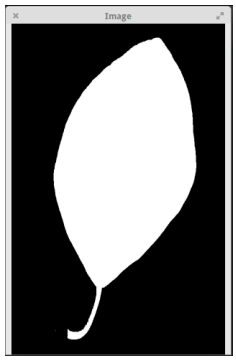

Figura 10: Aplicação de operação morfológica ”close”

E. *Encontrar e desenhar contornos*

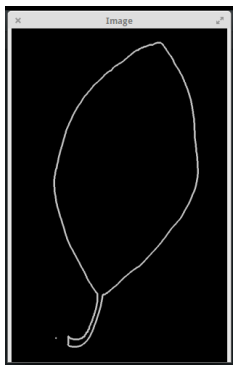

Figura 11: Exemplo de imagem após o drawContourns()

Após obtermos a imagem totalmente binarizada e sem ruído aparente é utilizado o método `findContours` que

permite encontrar os contornos, de seguida estes pontos são desenhados numa matriz preenchida previamente a zeros. O código utilizado neste contexto encontra-se de seguida.

```
mask=zeros(imgray.shape[:2],
uint8)          image,contours, hierarchy
= cv2.findContours(thresh,1,2)
cv2.drawContours(mask,contours,-1,
(200,200,0),3)
```

F. Extração de características

Após a segmentação são extraídas as características. O código referente à extração das mesmas pode ser encontrado no ficheiro `FeatureExtractors.py` e `Utils.py`.

F.1 Número de cantos

Para extrair o número de cantos usámos uma variação da função do algoritmo *Harris Corner Detection* usando a função do opencv `cv2.goodFeaturesToTrack()` que encontra os N cantos "mais fortes"numa imagem em grayscale. O excerto de código utilizado apresenta-se de seguida.

Listing 1: Função de extração de número de cantos

```
def get_corner_points(img, maxFeat):

  feature_params = dict( maxCorners = maxFeat,
      qualityLevel = 0.6, minDistance = 7,
      blockSize = 7 )
  corners = cv2.goodFeaturesToTrack(img, mask =
      None, **feature_params)
  return corners
```

F.2 Razão entre comprimento e altura

O comprimento por si só, numa imagem ou a altura/largura não nos dá grande informação numa imagem. Mas se juntarmos as duas e obtivermos uma razão entre elas podemos compreender que folhas são mais finas e longas ou mais largas e curtas, por exemplo, conseguindo distinguir assim algumas espécies através desta caraterística.

Listing 2: Função de extração da razão entre comprimento e altura

```
def length_width_ratio_feature_extractor(self,
    image):
    ratio = 0

    x_diff = max_x_diff(image)
    y_diff = max_y_diff(image)

    if y_diff > x_diff:
      ratio = y_diff/x_diff
    else:
      ratio = x_diff/y_diff

    features = array([ratio])
    feature_names =
        array(['length_width_ratio'])

    return (feature_names, features)
```

F.3 Razão entre perímetro e área

Há semelhança da caraterística anterior a razão entre o perímetro e área também é uma das características que encontrámos na literatura consultada.

Listing 3: Função de extração da razão entre perímetro e área

```
def perimeter_area_ratio_feature_extractor(self,
    image):

    area_points = get_image_area(image)
    edge_points = get_edge_points(image)
    perimeter = len(edge_points)

    perimeter_area_ratio = 1.0 * perimeter /
        area_points

    features = array([perimeter_area_ratio])
    feature_names =
        array(['perimeter_area_ratio'])

    return (feature_names, features)
```

F.4 Razão entre áreas

Na extração desta caraterística a imagem já chega com processamento anterior realizado, que será detalhado na seguinte secção do relatório. A imagem que chega aquando da extração da razão de áreas será algo semelhante a:

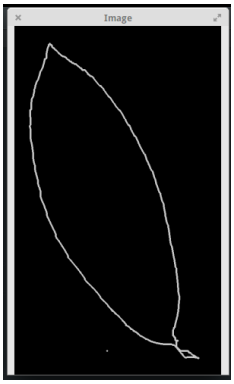

Figura 12: Exemplo de imagem após o pré-processamento inicial

A razão de áreas aqui explicitada é a razão entre a área dentro do contorno e a área fora.

Listing 4: Função de extração da razão enre áreas

```
def ratio_of_areas_feature_extractor(self,
    image):
  area_points = get_image_area(image)
  x_diff = max_x_diff(image)
  y_diff = max_y_diff(image)

  area_square = x_diff * y_diff

  ratio_of_areas = 1.0* area_square /
      area_points

  features = array([ratio_of_areas])
  feature_names = array(['ratio_of_areas'])

  return (feature_names, features)
```

F.5 Solidez

A solidez é calculada através da função seguinte, recebendo como input uma imagem semelhante à apresentada na figura 12.

Listing 5: Função de extração da solidez

```
def get_solidity(img):
  cnt = get_cnt(img)
  area = cv2.contourArea(cnt)
  hull = cv2.convexHull(cnt)
  hull_area = cv2.contourArea(hull)
  if hull_area == 0:
    hull_area = 0.0001
  solidity = float(area)/hull_area

  return solidity
```

F.6 Momentos

Tal como descrito anteriormente os momentos de uma imagem ajudam-nos a calcular algumas características como o centro de massa do objeto, áreas do objeto e informações sobre a sua orientação. O excerto de código seguinte permite-nos calcular os momentos numa imagem.

Listing 6: Função de extração dos momentos

```
def hu_momments_feature_extractor(self, image):
  feature_names = array(['hu1', 'hu2','hu3',
      'hu4','hu5', 'hu6','hu7'])
  moments = cv2.moments(image)
  hu_moments = cv2.HuMoments(moments).flatten()

  return (feature_names, hu_moments)
```

G. Classificadores

G.1 SVM

SVM[3] são modelos de aprendizagem com algoritmos que analisam dados utilizados para a classificação e análise de regressão. Dado um conjunto de exemplos de treino, cada um marcado como pertencente a uma categoria, um algoritmo de treino SVM constrói um modelo em que atribui a novos exemplos uma ou outra categoria, tornando-se num classificador linear binário não probabilístico.

Optámos assim por usar SVM visto que foi um classificador falado nas aulas práticas e adequado para o problema em questão.

H. Algoritmo de classificação

Após a recolha das características das imagens a testar, tal como foi referido anteriormente foi utilizado um classifacador SVM, disponível pelo sklearn em python. Para o treino foi utilizado o método `fit()` e para a previsão o método `predict()`. Após a execução do `predict()` será disponibilizada uma previsão para cada imagem de entrada que contém as características mais semelhantes.

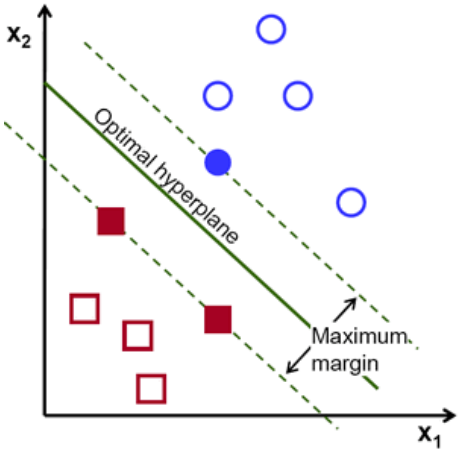

Figura 13: Exemplo visual do hiperplano para um SVM treinado com amostras de duas classes

V. EXECUÇÃO DO PROJETO

Os requisitos deste projeto são:

- Python versão 2.6 ou superior. https://www.python.org/downloads/
- OpenCV versão 3.0+ em Linux http://www.pyimagesearch.com/2015/06/22/install-opencv-3-0-and-python-2-7-on-ubuntu/
- Sklearn: http://scikit-learn.org/stable/install.html
- PyQt4: instalação windows ou em Linux repositório apt: `sudo apt-get install python-qt4`

O projeto requer um treino a priori que pode ser executado colocando as imagens binárias das folhas a treinar dentro da pasta *Data/DataTraining/*. Dentro desta pasta terá que existir outras que contenham o nome da folha que pretendemos treinar e respetivas imagens no seu interior. O programa em python `training_Leaves.py` irá extrair todas as características abordadas neste relatório e será criado um ficheiro CSV com todas essas. Este ficheiro de output será colocado na pasta `Data/Training_Tables`.

Para executar o projeto tiremos que executar `python test_Leaves.py` ou `python test_Leaves.py all` a diferença entre estes é que o segundo engloba a caraterística "Moments". Resultados das duas corridas serão analisados mais à frente.

Após isso é só esperar a extração das caraterísticas na imagem a analisar e que o classificador SVM compare e preveja o grupo à qual a imagem a analisar tem mais parecenças. Se tudo correr bem e o PyQt estiver instalado na máquina a imagem a analisar/input e uma previsão binária com a qual o classificador foi treinado irão aparecer no ecrã lado a lado, vendo o utilizador com mais facilidade a previsão obtida.

[3] Support Vector Machine

VI. RESULTADOS

Nesta secção iremos apresentar os resultados obtidos na realização deste projeto. Atualmente temos 10 tipos de folha no nossa pasta `Data/DataTraining` são eles:

- `Acer_Circinatum`
- `Castanea_Sativa`
- `Ginkgo_Biloba`
- `Liquidambar_Styraciflua`
- `Quercus_Shumardii`
- `Tilia_Tomentosa`
- `Acer_platanoides`
- `Ilex_Aquifolium`
- `Olea_Europaea`
- `Sorbus aucuparia`

Cada uma destas pastas têm cerca de 16 imagens binárias dessa espécie. Temos também um script em python que nos permite gerar este estas imagens binárias quando necessário. No total temos aproximadamente 160 imagens para extração de características e respetiva geração do ficheiro CSV de output. Para as imagens de entrada que temos, o processamento demora algum tempo, aproximadamente 15 minutos. Obtemos na literatura que este tipo de processamento para alguma quantidade de imagens pode demorar até vários dias ou até mesmo semanas.

Tal como descrito anteriormente é possível gerar dois tipos de ficheiros de saída:

- `output_nm.csv`: onde consta a extração de todas as características exceto dos momentos para todas as imagens de entrada.
- `output_all.csv`: onde consta a extração de todas as características para todas as imagens de entrada.

Atualmente, o bom resultado deste software depende das imagens que são treinadas e das usadas nos testes. Se uma imagem possuir um *background* com bastante ruído os resultados serão adulterados e não surgirá o resultado esperado.

No caso da utilização do ficheiro CSV que contém os momentos (característica) o resultado com o classificado SVM é piorado invés da utilização sem estes. Prevemos que caso fosse utilizado outro tipo de classificador (e.g redes neuronais) este seria muito mais assertivo e teria uma percentagem de acerto superior.

A imagem seguinte apresenta a interface gráfica resultante da execução do ficheiro `test_Leaves.py`.

Listing 7: Output no terminal

```
++++++++++++++++++Parameters++++++++++++++++++
Extracting Features without momments.
++++++++++++++++++++++++++++++++++++++++++++++
Extracting features in image
Extracting features in image
train...
predict...
+++++++ PrevisoesTest_Leaves +++++++
Test_Leaves/l10nr003.jpg: Sorbus aucuparia
Test_Leaves/Prays_oleae_1 (copy).jpg:
    Olea_Europaea
```

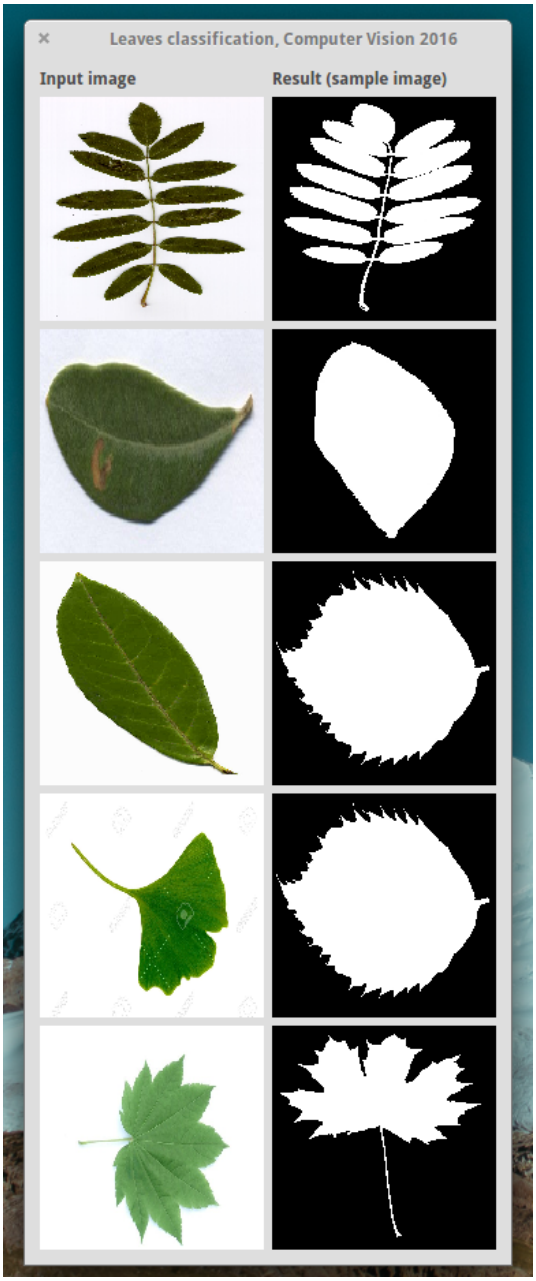

Figura 14: Exemplo de resultado final

VII. REPOSITÓRIO

O nosso código fonte está disponivel através de um repositório (GitHub) criado para o efeito.

https://github.com/ruipoliveira/leaves-classification-opencv

VIII. CONCLUSÕES

Chegado ao final deste relatório, é nossa intenção efetuar uma retrospetiva da evolução do mesmo, tendo em conta os problemas com que nos deparámos, e principais conclusões retiradas. Analisando os resultados obtidos podemos concluir que nem sempre os resultados nos deram previsões totalmente corretas da espécie em avaliação, já que estas previsões estão maioritariamente dependentes de um pré-processamento das imagens. Embora os resultados obtidos para as percentagens de acerto não sejam significativas, pensamos que as características extraídas das imagens são adequadas para a resolução do problema proposto. No entanto, estamos conscientes de que os objetivos propostos para o trabalho foram alcançados e, mais ainda, este trabalho permitiu-nos aprofundar os nossos conhecimentos sobre o reconhecimento e classificação de imagem, utilizando openCV. Adicionalmente, a utilização do QT em python foi uma mais valia para os conhecimentos adquiridos durante a realização do projecto, já que nunca o tínhamos utilizado em circunstâncias anteriores.

Apesar de termos alcançado os objetivos propostos para este trabalho, existem alguns pontos que consideramos que possam ser melhorados futuramente, dos quais destacamos os seguintes:

- Usar *deep learning* ou redes neuronais para aumentar a percentagem de acertos.
- Aumentar/melhorar o número de características extraídas (e.g trabalhar com a nervura das folhas).
- Melhorar a previsão para imagens em contexto real, tentando fazer tracking da folha a analisar na imagem e posteriormente realizar a sua segmentação/processamento.
- Aumentar os dados de treino, permitindo utilizar também imagens coloridas.

REFERÊNCIAS

[1] Neves, A. J. R.; Dias, P. Slides teóricos Visão por Computador (2016)

[2] Documentação OpenCV. hhttp://docs.opencv.org/. Web. 15 Outubro 2016.

[3] Silva, Pedro Filipe Barros "Development of a System for Automatic Plant Species Recognition". Disponível para leitura e download em https://repositorio-aberto.up.pt/handle/10216/67734. 2013.

[4] SVM Classifier. http://jmgomez.me/a-fruit-image-classifier-with-python-and-simplecv/ 19 May 2014.