# Flower detection using features

Computer Vision, UPC

Xavier Martín Ballesteros and Adrià Cabeza Sant'Anna
UNIVERSITAT POLITÈCNICA DE CATALUNYA

June 9, 2019

# Contents

# 1 Introduction

The aim of this assignment is to classify 12 different types of flowers using feature extraction.

- BotodOr
- Crocus
- Fadrins
- Gerbera
- Hemerocallis
- Narcis
- Buixol
- DentdeLleo
- Fritillaria
- Girasol
- Lliri
- Viola

We have first implemented several ways to extract features that we believed could be key in order to classify correctly the flowers, we tested them, and using different classifiers (decisions trees, SVM or random forest) have tested the accuracy, specificity or sensitivity.

# 2 Descriptors

In the following sections we will introduce different descriptors we have used to find those valuable features.

## 2.1 Compactness

After checking the different species of flowers we have to detect, one of the first descriptors we came up with was the **compactness**, the ratio of the perimeter to the area of the region. Since the shape and the size of each flower really varies, we think it can be a good discriminator.

To implement it we do need two things: **perimeter**($P$) and **area**($A$).

$$C = \frac{(P^2/Area)}{4 * \pi}$$

To get the area we have used Matlab's **regionprops** utility, and to get the perimeter we have made **an erosion and a substraction**.

In this example, we compare a *Hemerocallis* and a *Botó d'Or*. Since the measure takes a minimum value of 1 for a circle and objects which have an elliptical shape, or boundary that is irregular rather than smooth, will have larger compactness. We should have smaller values for the *Botó d'Or*.

(a) Compactness: **28.10**

(b) Compactness: **0.78**

Figure 1: Comparison of compactness

## 2.2 Color

Most of the flowers, show always the same one or two colors. So the color is really important in order to distinguish an specie from another. To do it we have used the Mathworks utiliy: **image2palette which uses k-means color clustering**. We executed the k-means with 4 clusters (one is for the black which is always present) to get the most important colors; moreover, we have also the information of the presence percentage of each color.

This is an example with a *Viola* picture:

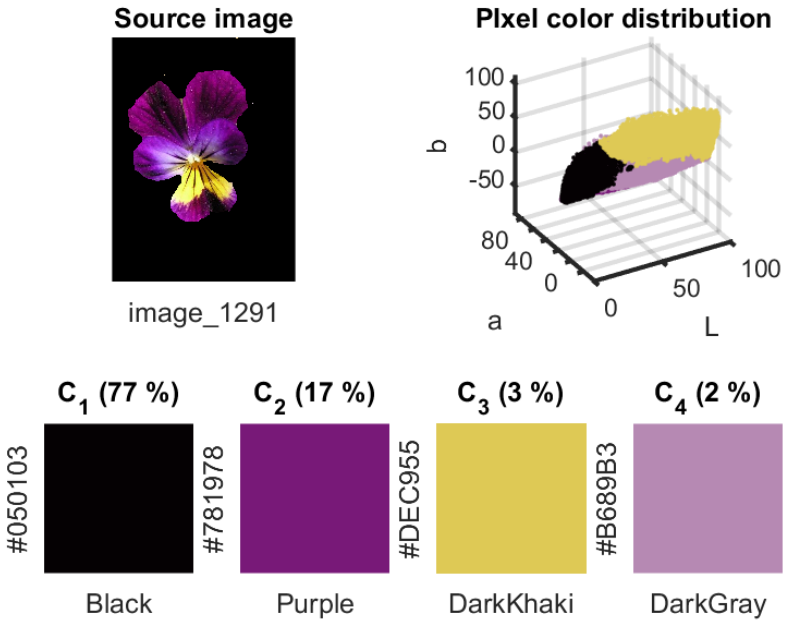



Figure 2: Example of our color feature extraction

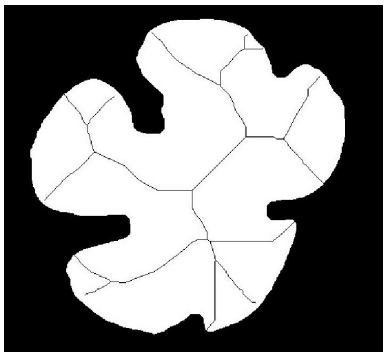
## 2.3 Number of petals

This descriptor was one of the most difficult to think of. It can be done in several ways so we first had to choose in which way we wanted to attack the problem. In our case we decided to **skeletonize the flower**.

To do so we used **bwmorph(image, 'skel', inf)** from Matlab. This creates the skeleton iterating as many times as necessary in order to not see any change between iterations (until it converges). Then we observed that sometimes the skeleton did not touch the contourn of the flower so we could not really count the petals; to solve it we applied an small erosion. The value of the disk structure was chosen based on several trials.

In this example using *Boto d'Or* we get as a result 5 petals:

(a) Original



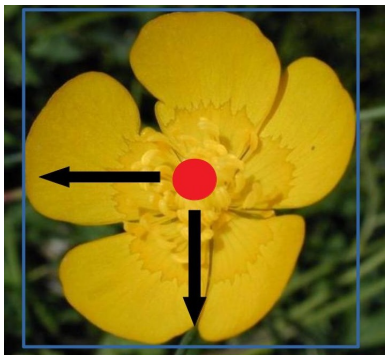
(b) Skez of the image

## 2.4 Relative position of the centroid

Just evaluating the position of the centroid would have been useless, it is not a feature of the flower per se. It can vary depending on the angle of the photo or the distance. However, if we calculate the relative position we can get really powerful information.

This two flowers, for example, should have really different values for this descriptor and it could be key in order to differenciate them in our classifier:

(a) Centroid and bounding box of a *Hemerocallis*



(b) Centroid and bounding box of a *Girasol*

The way our descriptor is implemented is the following: we get the first and last pixels of X and Y axis in order to create the **bounding box**, then using **regions props** we get the **centroid of the flower** and finally we calculate the **relative position** of the centroid over the bounding box.

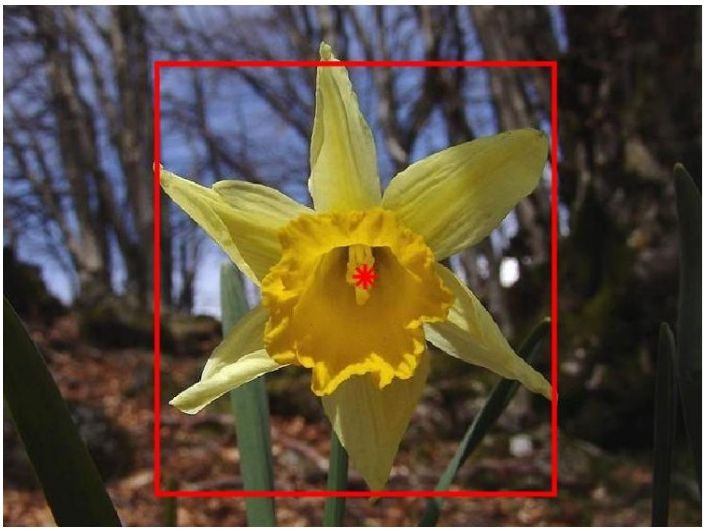
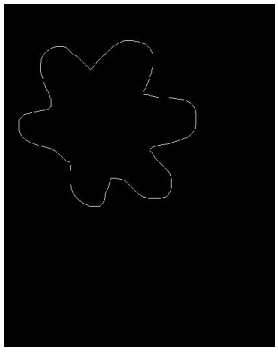


Figure 5: Example of centroid and bounding box

## 2.5 Fourier descriptor: Shape

It is known that we can use the Fourier transform to analyse and characterize the shape of a boundary. Since there are some flowers like the *Hemerocallis* or the *Girasol* that have a very specific shape; we believe that **the shape** can be a great descriptor.

(a) Original



(b) Shape

Figure 6: Fourier descriptors

## 2.6 Hogs form: Orientation

Previously we have introduced a way to describe information about the shape. However, there are another ways to describe shapes.
As we wanted to have as proper features as possible, we have also implemented HOGs (histogram of oriented gradients), which basically describes **the shapes as the distribution of intensity**

**gradient and edge directions**. To do it, we have used the utilities available at the *Matlab Computer Vision Toolbox*.

(a) Original

(b) Hog features

Figure 7: HOG descriptors

# 3 Classifiers

# 4 Experiments

# 5 The *zero* class

In order to increase the precision of our model, we have added a new class called *zero*. This class represents all the flowers that are not from any of the 12 species that we are observing.

To implement it we have written a script that **crawls google images and downloads photos of other species of flowers**. This will be very useful to create our dataset of *zero* flowers. Morover, since these images are not segmented, we have looked for another source of flower pictures. In this case we have used **the Flowers Dataset from the Visual Geometry Group** (University of Oxford). This approach will help us to save time because we will have some flowers segmented by us (reviewed manually afterwards) and some already segmented (which we suppose are already correct).

Our approach will consists in not training our classifier with that photos: we will assign a picture to that class if the confidence is less than a threshold. To test if actually works we will use the already mentioned test.

Some of the flowers that can be seen in our *zero* flower dataset are:

(a) Azalea

(b) Roses

(a) Gazania

(b) Tulip

Figure 9: Zero class flowers

# 6 Data augmentation

In order **to strenghten our descriptors** we have also used the following public repository: *Albumentation: fast image augmentation library and easy to use wrapper around other libraries*(see more in References), which proporcionates facilities to augment the dataset including several transformations like: flipping, blurrring, RGB Shifting, Random contrast, Random brightness, etc...

To decide which transformations we wanted to apply, we looked first to our descriptors and the importance of each feature. For example, we discarted the Channel Shuffle because we believe that the color is very important for our implementation. Then, based on trial and error, we picked some of them which gave us an overall better result. Finally we are using random variations of:

- Brightness and Contrast
- Blur
- Rotations
- Flips

For example let's suppose that we are working with this *Boto d'Or* image:

Figure 10: Original picture

The script we have implemented would apply the following transformations:

(a) Blur transformation

(b) Horizontal Flip transformation

(a) Brightness and Contrast transformation

(b) Scale transformation

Figure 12: Data augmentation transformations

# 7 Results

# 8 List of functions and libraries that have been used

- **Bounding Box**: function that finds the relative position of the centroid from the bounding box. It uses the matlab utility **regionprops** to get the centroid.
- **getCompactness**: function that calculates the compactness of a flower. It uses the matlab utility **regionprops** to get the area. To get the perimeter we have made an erosion and then a substraction.
- **getForma**: function that gets the shape Fourier descriptor.
- **getHog**: function that gets the HOG features. It uses the **Matlab Computer Vision Toolbox** utility **extractHOGFeatures**.
- **getNumberPetals**: function that gets the number of petals using the skeleton. It uses the matlab utility **bwmorph** for the skeletonization.
- **NAMEOFTHEFUNCTION**: function that gets the 3 main colors of the image using k-means and its presence percentage. It uses the **Mathwork: image2palette** utility which clusters pixels colors using k-means to make a palette.
- **Albumentations library**: for data augmentation issues.

- **Google images download library**: to download several pictures of the *zero* class.

# References


[1] *Image Segmentation with MATLAB* [online]. Available at : https://www.mathworks.com/discovery/image-segmentation.html [Accessed 28 May 2019]

[2] Nilsback, M-E. and Zisserman, A. (2008) *Automated Flower Classification over a Large Number of Classes* [online] Available at:http://www.robots.ox.ac.uk/ vgg/publications/papers/nilsback08.pdf [Accessed 26 May 2019]

[3] M.A. Wirth.(2004) *Shape Analysis & Measurement* [online] Available at: http://www.cyto.purdue.edu/cdroms/micro2/content/education/wirth10.pdf [Accessed 24 May 2019].

[4] Buslaev A.(2018). Based on *Albumentations: fast and flexible image augmentations* [online]. Paper available at:https://arxiv.org/abs/1809.06839. Code available at: https://github.com/albu/albumentations [Accessed 3 June 2019].

[5] Vasa H. (2019). *Python Script to download hundreds of images from 'Google Images'.* [online] Code available at: https://github.com/hardikvasa/google-images-download [Accessed 4 June 2019].