ICT for Health
Laboratory # 2
Moles

Monica Visintin

Politecnico di Torino

ICT4SS
Information and Communications Technologies for Smart Societies

2018/19

Table of Contents

The data [1]

- Download file images.zip from the folder materiale of the class
- Unzip the file, you'll get several jpeg images of moles, the name of the files are
  1. low_risk_n.jpg (where n is an integer) for moles that have a low probability of being melanoma (i.e. tumors)
  2. medium_risk_n.jpg (where n is an integer) for moles that have a low probability of being melanoma
  3. melanoma_n.jpg (where n is an integer) for moles that have a high probability of being melanoma
- View all the pictures, so that you have an idea of what you have to work with.

Goal of the lab

- We want to help medical doctors in the analysis of the moles
- 5 features are considered by the doctor to diagnose melanoma: ABCDE

  A asymmetry
  B border
  C color
  D diameter
  E evolution

- We want to analyze **borders** and get **a feature** that will be used, together with other features regarding asymmetry, color, etc, to classify moles. Other researchers will define these other features, we will just work on borders.

Main idea

- We use K-means in scikit-learn to find three clusters (quantization of the image with three levels of colour)
- We find the contour of the darkest cluster, corresponding to the mole
- We evaluate the area of the cluster corresponding to the mole and the length of the contour (perimeter of the mole)
- We evaluate the perimeter of a perfect circle with area equal to that of the mole
- We evaluate the ratio between the perimeter of the mole and the perimeter of the corresponding circle: the higher is this value the more **indented** is the border.

The jpeg image [1]

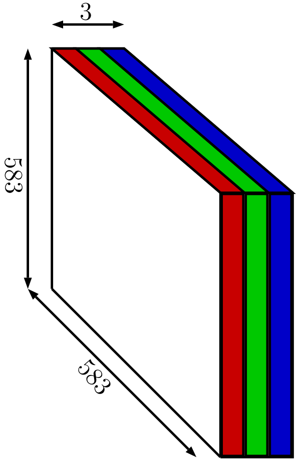
- The jpeg image is an image that has been compressed according to the jpeg standard
- To read the image in Python:

```
import matplotlib.image as mpimg
filein=...
im = mpimg.imread(filein)
```

- `im` is an Ndarray with shape **583 x 583 x 3** and elements of type unint8 (unsigned integer with 8 bits); the image is made of 583 x 583 pixels
- `im[:,:,0]` stores the amount of **red** color, from 0 to 255
- `im[:,:,1]` stores the amount of **green** color, from 0 to 255
- `im[:,:,2]` stores the amount of **blue** color, from 0 to 255
- Value `[0,0,0]` corresponds to black, value `[255,255,255]` corresponds to white

The jpeg image [2]

the image dimension is 3



The jpeg image [3]

- To show the image in Python:

```
plt.figure()
plt.imshow(im)
plt.title(’original image’)
plt.show()
```

- To **force** Python to show the image before the end of the execution of the script:

```
plt.figure()
plt.imshow(im)
plt.title(’original image’)
plt.pause(0.1)
```

This method is deprecated, but it works until they fix the bug...

K-means [1]

- Import K-means by writing:

```
from sklearn.cluster import KMeans
```

- To instantiate the K-means object write:

```
kmeans = KMeans(n_clusters=3, random_state=0)
```

- To find the clusters you should write:
  `kmeans.fit(im)`
  but Python gives you an error, because it requires a 2D Ndarray, not a 3D Ndarray

K-means [2]

- What can we do? the k-means algorithm does not take into consideration the order of the data, therefore we can reshape the 3D Ndarray into a 2D Ndarray:

```
[N1,N2,N3]=im.shape
im_2D=im_or.reshape((N1*N2,N3))# N1*N2 rows and N3 columns
```

- Then, use k-means as:

```
kmeans.fit(im_2D)
```

- Note that class KMeans takes time to find the clustering: it actually tests some hundreds of different initial vectors and gives you the best clustering that minimizes the moment of inertia

K-means [3]

- The attributes of class `KMeans` are:

`kmeans.cluster_centers_`, the centroids of the clusters
`kmeans.labels_`, the `N1*N2` classes/clusters each pixel belongs to

Note that the centroids are float numbers, but we need `uint8` numbers to show the image; therefore the centroids become:

```
centroids=kmeans.cluster_centers_.astype('uint8')
```

These three centroids represent the three colors that k-means found as representatives of all the image colors (the original image has potentially $2^{24} \simeq 16 \times 10^6$ different colors, but we want only 3 different colors)

- **Generate the image with only 3 colors, plot it and look at the difference between the original and the quantized image.**

The contour [1]

- The classical algorithm to find the contour is the “**snake**” algorithm (or active contour), available both in Matlab and Python. **You are NOT allowed to use the snake algorithm in this lab** (otherwise you have nothing to do....)
- The idea is the following:
  1. Among the centroids, find the darkest color, it is the colour of the mole (the other two colours are for the skin and shadows, typically)
  2. Find the median (not the mean) of the region where the quantized image is equal to this darkest color: this median is probably the center of the mole or it is close to be the center of the mole; note that other (small?) portions of the image might have the same darkest color, so this task might not be so easy.
  3. Implement an algorithm that, starting from the center of the mole, finds a square or rectangular region that includes all the mole, but not the other areas (on the borders of the original image, typically) with the same color

The contour [2]

4. Consider this sub-image and **invent an algorithm to find the contour**. **You are not allowed to discuss this specific topic with the other students, I expect that each student will find a different solution**
5. Plot the contour you found. Matplotlib class `matshow` might be better than `imshow`; note that the input of `matshow` must be an NDarray of dimension 2, not an image; you can specify the color map giving `matshow` the optional parameter `cmap='Blues'` (or other mappings); you can add the colorbar by writing `plt.colorbar()` right after `plt.matshow()`. See the help of `matshow`
6. Once you have your contour find the ratio between the perimeter of the mole and the perimeter of the circle with the same area, analyze all the images and write a table with these ratios; include this table in the report.

- **Report due by December 31st 2018.**