Homework **before Practical 2**

Credit: 2 points. Upload your codes to Dropbox (Practical2 subfolder)
by **Wednesday, October 9, 11;59 PM**

(Jin Huang, Adam Czajka; initial version: 01/28/2016; modified: 10/03/2019)

1 Prepare your laptop for Practical 2

This practical will again use Python. Before starting, we need to install a package called `scikit-image`. You can use the following command to install this package easily:

- If you are using Anaconda:
  `conda install -c anaconda scikit-image`
- Or you can choose to use pip:
  `pip install scikit-image`

You can find the tutorial and documents for this package from their website: https://scikit-image.org

2 What we will be doing in class on Friday?

Our final goal in this practical session is to classify different simple objects in the image provided in class. The procedure for object classification in this example will follow the general pipeline in Computer Vision:

```
        Image
          |
          v
 .---------------.
 | Preprocessing |
 '---------------'
          |
          v
 .--------------.
 | Segmentation |
 '--------------'
          |
          v
  .-------------.
  | Feature     |
  | extraction  |
  '-------------'
          |
          v
 .----------------.
 | Classification |
 '----------------'
```

2.1 Pre-processing

The first step involves thresholding and converting the image to a binary image, as we did in Practical 1. The result will be a binary image with each object of interest expressed as a connected component of white pixels. Ideally, each connected component is an object to classify. However, in practice, objects may be only partially binarized, or they may overlap. Intensity of images that you will process in class may have some linear trend, thus its removal should help in getting a better binary image. You may also see the need for histogram equalization before binarization.

2.2 Segmentation

Assuming that we already have a good binary image, your next task will be to find all the white blobs representing objects of interest. `scikit-image` package provides a very nice command to find all the connected components:

```
from skimage import measure

labels = measure.label(bw_img, n)
```

in which:

- `bw_img`: binary image
- `n`: connectivity

`n` can take values of 4 or 8, indicating either 4-connected or 8-connected.

2.3 Feature extraction

In this practical we will use simple geometrical features to classify several classes of objects. For instance, looking for elliptic shapes we can model each localized object as an ellipse. The ratio of major axis to minor axis of the localized objects should be greater than 1 for ellipses, while for, say, round objects the ratio should be roughly equal to 1.

This example illustrates that even trivial features such as major-to-minor axes can help to classify simple objects. In this part, you will use a very useful `scikit-image` function `regionprops` to extract basic features from the connected components:

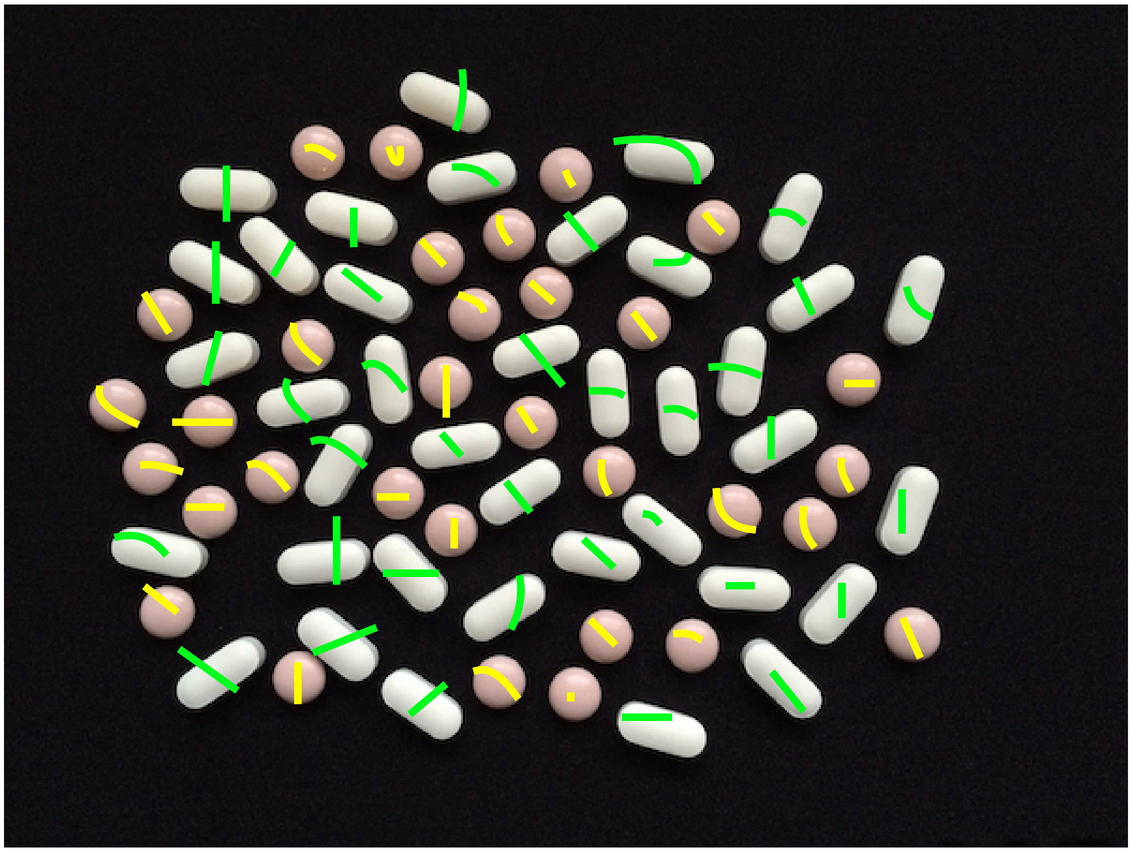
```
features = measure.regionprops(labels)
```

where:

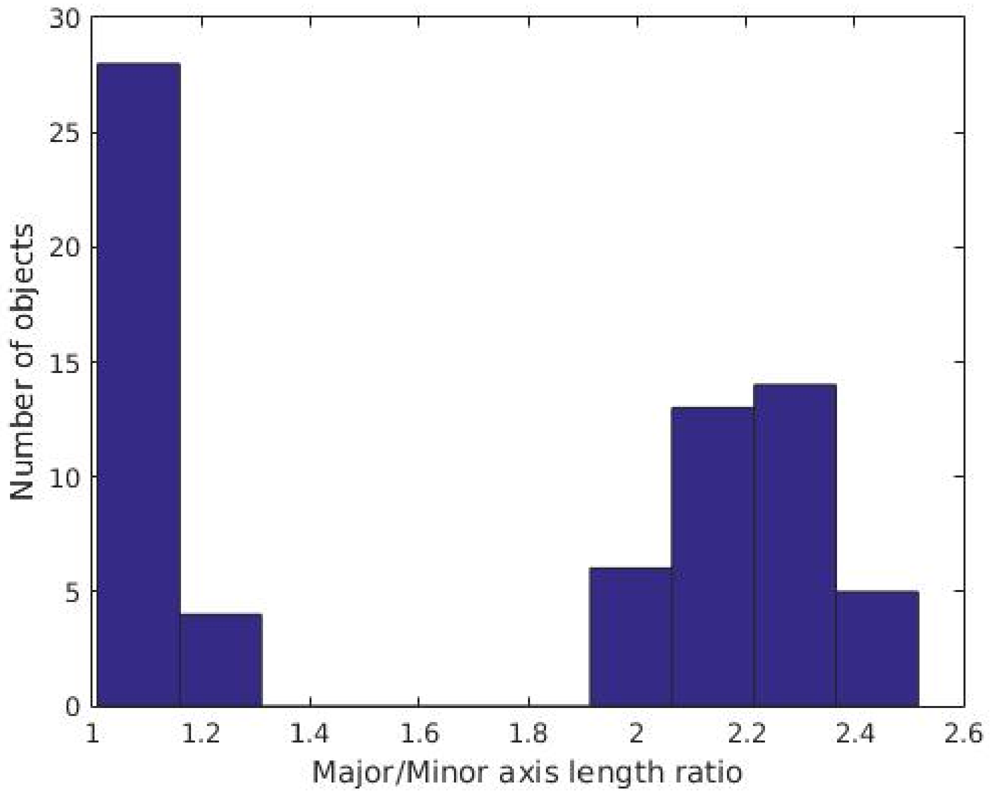
- `labels` = connected components,
- `features` = comma-separated list of properties we want to extract, for example: 'area', 'bbox', 'centroid', ... For a complete list of properties check `https://scikit-image.org/docs/dev/api/skimage.measure.html?highlight=regionprops#skimage.measure.regionprops`

2.4 Classification

For illustration, assume we want to distinguish oval pills from the round ones in this picture:



If we use `regionprops` to get major and minor axis length for each connected component, and plot the histogram of their ratio, we see that this single feature is sufficient to classify these shapes perfectly:



Certainly, one feature might not be enough to separate the classes. Using more features might lift up the data points to a higher-dimension feature space in which they are separated (see this nice illustration).

Now, when we have the features determined, we can build a simple classifier to classify the objects: just check if the ratio is above, say, 1.5. If it is, the object is *elliptical*. If it is not, the object is *round*.

3 Your task

Write a Python program that incorporates all steps described in Section 2 and count how many round and oval pills are in the example image (pills.png) attached to this document. Upload your codes to SAKAI Dropbox (Practical2 subfolder) by **Wednesday, October 9, 11;59 PM**. If the code works well, and the number of pills automatically reported by your program is correct, you will get 2 points for this task.