05-histograms

January 14, 2020

1 Creating Histograms

In general, as you might already know, a histogram is a basic method for representing a statistical distribution. As it pertains to images, a histogram is a graphical representation showing how frequently various color values occur in the image. If your project involves detecting color changes between images, histograms will prove to be very useful, and histograms are also quite handy as a preparatory step before performing thresholding.

1.0.1 Grayscale Histograms

```
[ ]: # import needed libraries
     import skimage
     import numpy as np
     from skimage.viewer import ImageViewer

     img = skimage.io.imread('../data/seedling.tif', as_gray = True)
```

To plot the histogram we make use of the standard scientific plotting libray in python: matplotlib. This also allows us to directly (but non-interactively) plot the image within the notebook.

```
[ ]: import matplotlib.pyplot as plt
     # 'magic' to display plots in the jupyter notebook
     %matplotlib inline

     plt.imshow(img, cmap = 'gray')
```

```
[ ]: # create the histogram
     histogram, bin_edges = np.histogram(img, bins=256, range=(0, 1))
```

```
[ ]: # plot the histogram
     f = plt.figure()
     plt.plot(bin_edges[:-1], histogram)
     plt.xlabel('grayscale value')
     plt.ylabel('counts')
```

1.0.2 Exercise: Mask the Seedling

The image contains largely the dark and uninformative background pixels. Similar to what we did with the maize roots, define and apply a rectangular mask around the seedling to limit our analysis to the region of interest. Then repeat the steps above to calculate and plot the histogram only for the pixels in the region of interest.

Hint: Use np.logical_not to invert the mask and select the pixel values inside the rectangle for the computation of the histogram.

[ ]:
```
%load ../exercises/05-MaskSeedling.py
```

[ ]:
```
mask = np.ones(image.shape, dtype = bool)
rr,cc = skimage.draw.rectangle(start =  (200,410), end = (400,500))
mask[rr,cc] = False
mask_i = np.logical_not(mask)

histogram, bin_edges = np.histogram(image[mask_i], bins=256, range=(0, 1))

plt.plot(bin_edges[:-1], histogram)
plt.xlabel('grayscale value')
plt.ylabel('counts')
```

1.0.3 Masking Pitfalls

[ ]:
```
# using a small toy matrix demonstrate difference of:
# image[mask_i] = 4 -> (in place array manipulation)
# flat = image[mask_i] -> extraction of unmasked elements into a flat array

img = np.ones( (4,4) ) * 7
img[1:3,:] = 3
print(f'image = \n{img}')
mask = img > 3
print(f'mask = \n{mask}')
```

[ ]:
```
img[~mask] = 99 # bit-wise logical not
print(img)
flat = img[~mask]
print(flat)
```

1.0.4 Color Histograms

We can also create histograms for full color images, in addition to grayscale histograms. For RGB images this will give three distinct distributions, one for each RGB component. The same strategy can also be applied if you have say two (or more) different fluorescent channels.

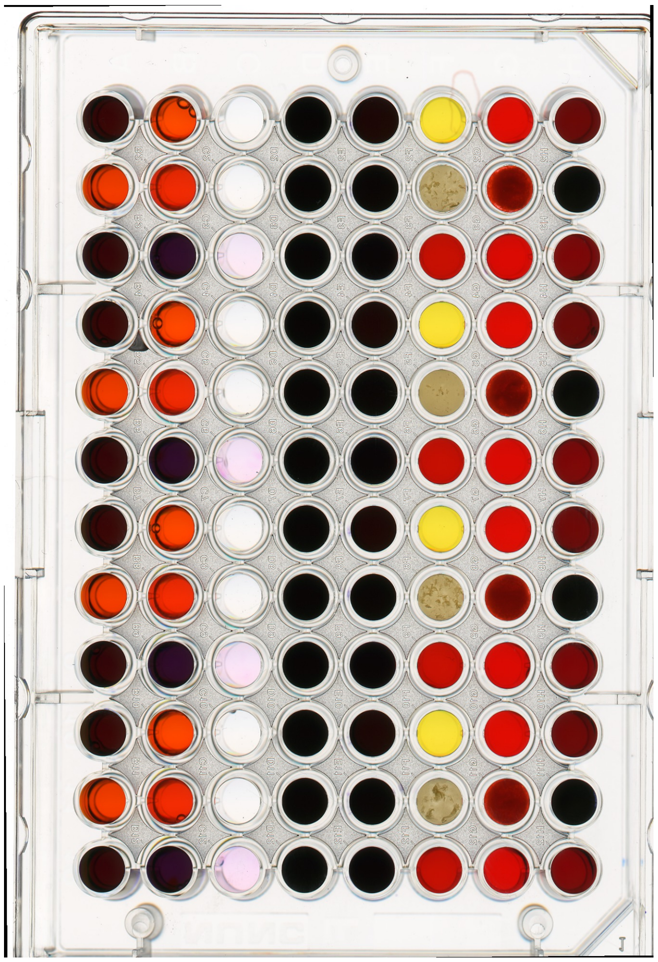
```
[ ]: imageRGB = skimage.io.imread('../data/seedling.tif')
     plt.imshow(imageRGB)
     plt.imshow(mask_i, alpha = 0.6, cmap = 'gray')
```

```
[ ]: # map channel names to array indices
     chan_to_ind = {'red' : 0, 'green' : 1, 'blue' : 2}

     plt.figure(figsize = (10,4))
     for channel_name in chan_to_ind:
         ind = chan_to_ind[channel_name]
         channel = imageRGB[...,ind]
         histogram, bin_edges = np.histogram(channel[mask_i], bins = 256, range =␣
      ↪(0,256))
         plt.plot(bin_edges[:-1], histogram, color = channel_name, label =␣
      ↪channel_name)

     plt.xlabel('Color value')
     plt.ylabel('Counts')
     plt.title('Color Histogram')
     plt.legend()
```

1.0.5 Exercise: Color Histogram with a Mask

We can also apply a mask to the images we apply the color histogram process to, in the same way we did for grayscale histograms. Consider this image of a well plate, where various chemical sensors have been applied to water and various concentrations of hydrochloric acid and sodium hydroxide: 

Suppose we are interested in the color histogram of one of the sensors in the well plate image, specifically, the seventh well from the left in the topmost row, which shows Erythrosin B reacting with water.

- As we did before, determine the center coordinates of the well of interest
- Use the appropriate skimage.draw method to define a circular region of interest
- Use this ROI to define an appropriate mask
- As an intermediate result plot the masked image, it should only be non-zero around the well we are interested in
- Use the circular mask to apply the color histogram operation to that well

[ ]:
```
%load ../exercises/05-ColorHistogramMask.py
```

1.0.6 Exercise: Histograms for the morphometrics challenge

Using the grayscale and color histogram programs we developed in this episode, create histograms for the bacteria colony images in the ../data directory: * colonies01.tif * colonies02.tif * colonies03.tif

Save the histograms for later use.

Hint: You might want to write two functions like 'plot_hist' and 'plot_histRGB' which take an input file path as argument and plot the respective histograms.

```
[ ]: def plot_hist(img_path):
         '''
         Plots the grayscale histogram of the image at *img_path*
         '''
         img = skimage.io.imread(img_path, as_gray = True)
         histogram, bin_edges = np.histogram(img, bins = 256, range = (0,1))

         plt.plot(bin_edges[:-1], histogram, color = 'k')
         plt.xlabel('Pixel value')
         plt.ylabel('Counts')
         plt.title('Grayscale Histogram')
```

```
[ ]: plot_hist('../data/colonies01.tif')
```

```
[ ]: def plot_histRGB(img_path):
         '''
         Plots the color histogram of the image at *img_path*
         '''
         # map channel names to array indices
         chan_to_ind = {'red' : 0, 'green' : 1, 'blue' : 2}

         img = skimage.io.imread(img_path)

         for channel in chan_to_ind:
             ind = chan_to_ind[channel]
             histogram, bin_edges = np.histogram(img[...,ind], bins = 256, range =␣
      ↪(0,256))

             plt.plot(bin_edges[:-1], histogram, color = channel, label = channel)

         plt.xlabel('Color value')
         plt.ylabel('Counts')
         plt.title('Color Histogram')
         plt.legend()
```

```
[ ]: plot_histRGB('../data/colonies02.tif')
```