

1 Python correction



```

1 from scipy import misc
2 import matplotlib.pyplot as plt
3 from skimage import exposure
4 import numpy as np
5 import sys

```

1.1 Intensity transformations

1.1.1 γ correction

The function skimage.exposure.adjust_gamma is used in order to adjust the gamma of the image. It is before normalized (values are float between 0 and 1). The results are illustrated in Fig.1.



```

I=misc.imread("osteoblaste.png")
2 I = I / np.max(I);

4 gamma=2;
I2= exposure.adjust_gamma(I, gamma);
6 plt.imshow(I2);
plt.show();

```

1.1.2 Contrast stretching

The LUT look-up-table is simply a function applied to the original image. In order to avoid division by zero, the smallest float value is introduced. Fig.2 illustrates the results.



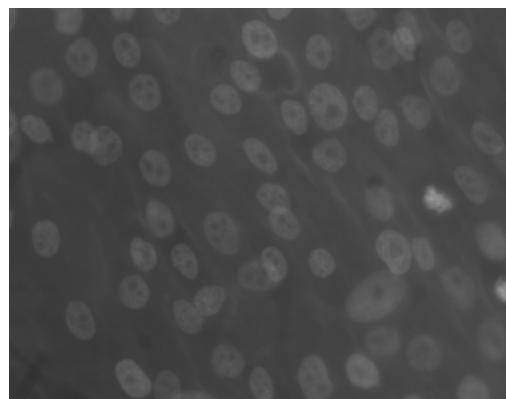
```

1 def contrast_stretching(I, E):
2     epsilon = sys.float_info.epsilon;
3     m = np.mean(I);
4     I = I.astype("float");
5     Ar = 1. / (1.+(m/(I+epsilon))**E);
    return Ar;

```

1.2 Histogram equalization

The histogram is displayed with the following function:



(a) Original image.

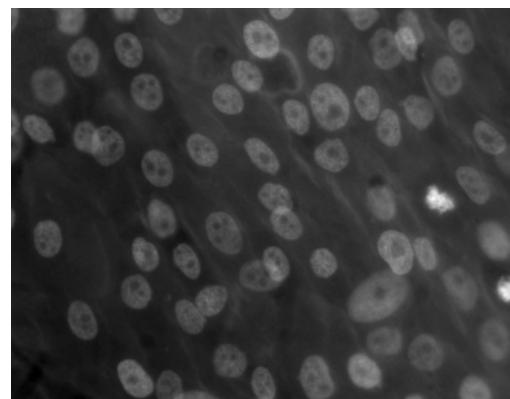
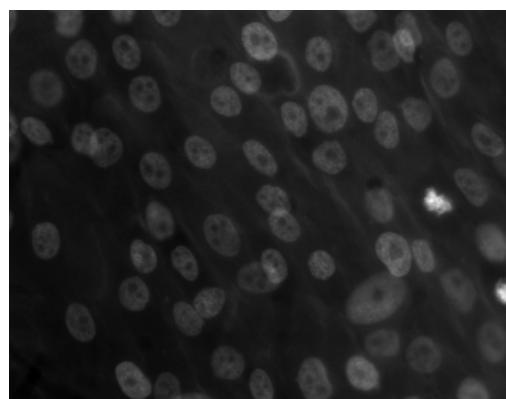
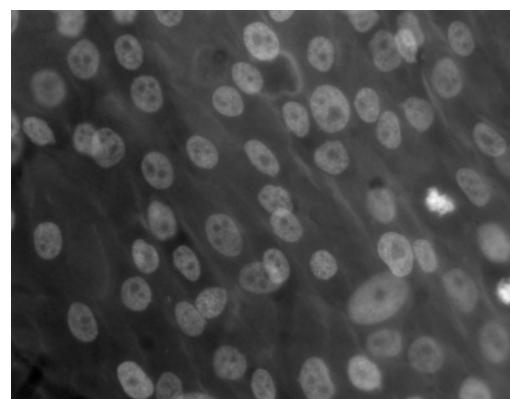
(b) $\gamma = 1$.(c) $\gamma = 2$.(d) $\gamma = 0.5$.

Figure 1: Gamma transforms.

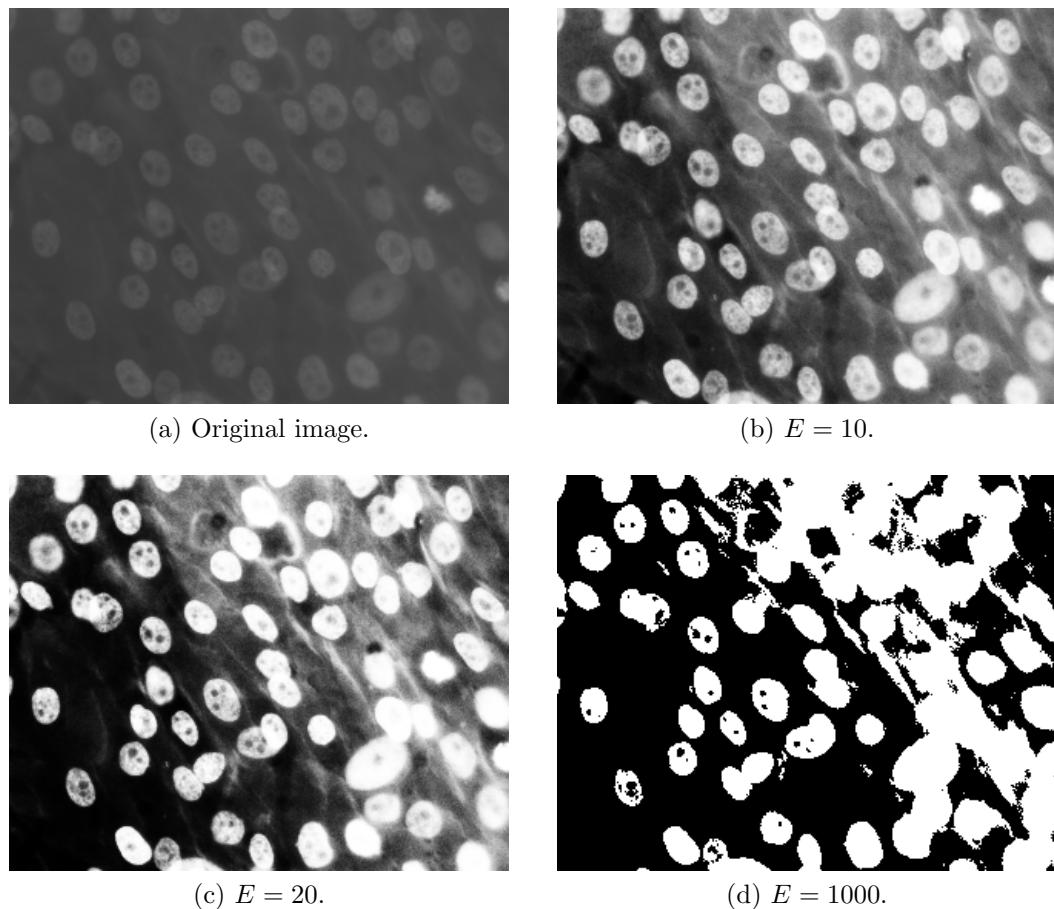


Figure 2: Constrain stretching.



```

1 def displaySaveHisto(I, filename=None):
2     """
3         Display and save pdf (if filename provided) of histogram of image I
4         If values are between 0 and 1, they are multiplied by 255
5     """
6     if np.max(I)<=1:
7         I = 255 * I;
8     hist,bins = np.histogram(I.flatten(),256,range=(0,255))
9     fig = plt.figure()
10    plt.bar(bins[:-1], hist, width=1);
11    plt.show();
12    if filename!=None:
13        fig.savefig(filename, bbox_inches='tight')

```

The histogram equalization is done via the skimage.exposure.equalize_hist function. The results are illustrated in Fig.3.



```

# histogram equalization
I2 = exposure.equalize_hist(I);
plt.imshow(I2);
plt.show()

```

The look-up-table (LUT) or cumulative distribution function (cdf) is computed and used in the next function:



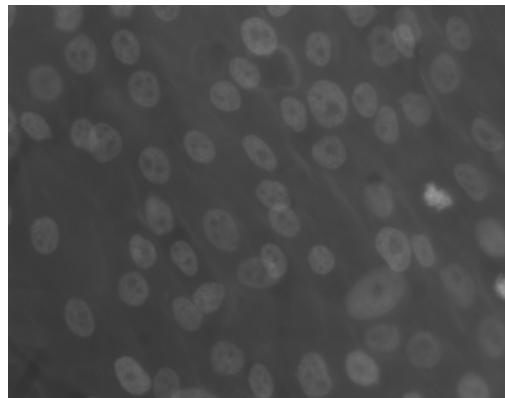
```

def histeq(I):
    """
    histogram equalization, version with look-up-table
    I: original image, with values in 8 bits integer
    """
    hist,bins=np.histogram(I.flatten(), 256, range=(0,255))
    cdf = hist.cumsum();
    cdf = (cdf / cdf[-1]);
    return cdf[I];

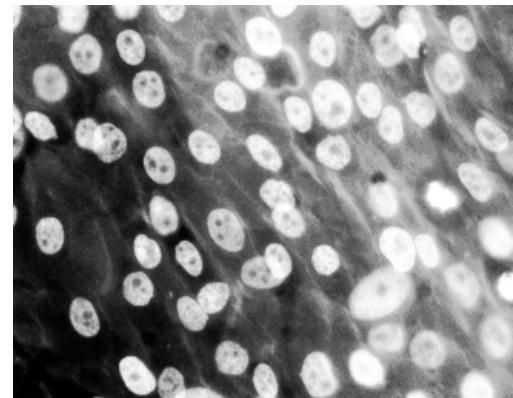
```

1.3 Histogram matching

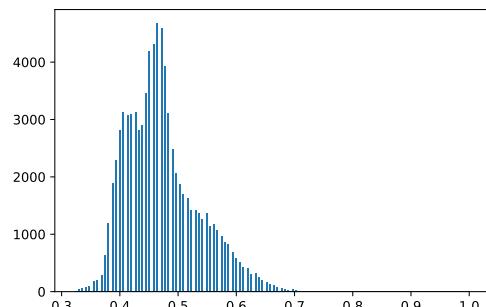
This method is only an approximate method. It requires an interpolation of the cumulative distribution functions in order to find the transformation. Results are shown in Fig.4



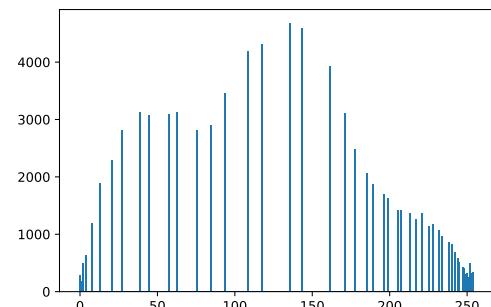
(a) Original image.



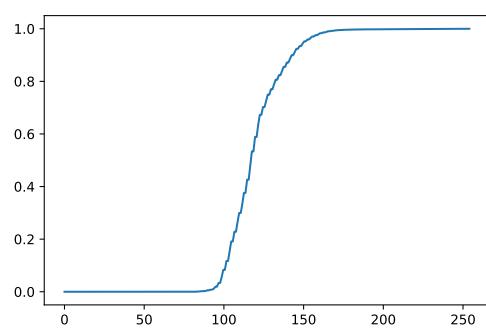
(b) Histogram equalization.



(c) Histogram of original image.



(d) Histogram after equalization.



(e) LUT (cumulative distribution function).

Figure 3: Histogram equalization.



```
def hist_matching(I, cdf_dest):
    """
    Histogram matching of image I, with cumulative histogram cdf_dest
    This should be normalized, between 0 and 1.

    This version uses interpolation
    """
    imhist,bins = np.histogram(I.flatten(),len(cdf_dest),density=True)
    cdf = imhist.cumsum() #cumulative distribution function
    cdf = (cdf / cdf[-1]) #normalize between 0 and 1

    # first: histogram equalization
    im2 = np.interp(I.flatten(),bins[:-1],cdf)

    # 2nd: reverse function
    im3 = np.interp(im2, cdf_dest, bins[:-1])

    # reshape into image
    imres = im3.reshape(I.shape)
    return imres;
```

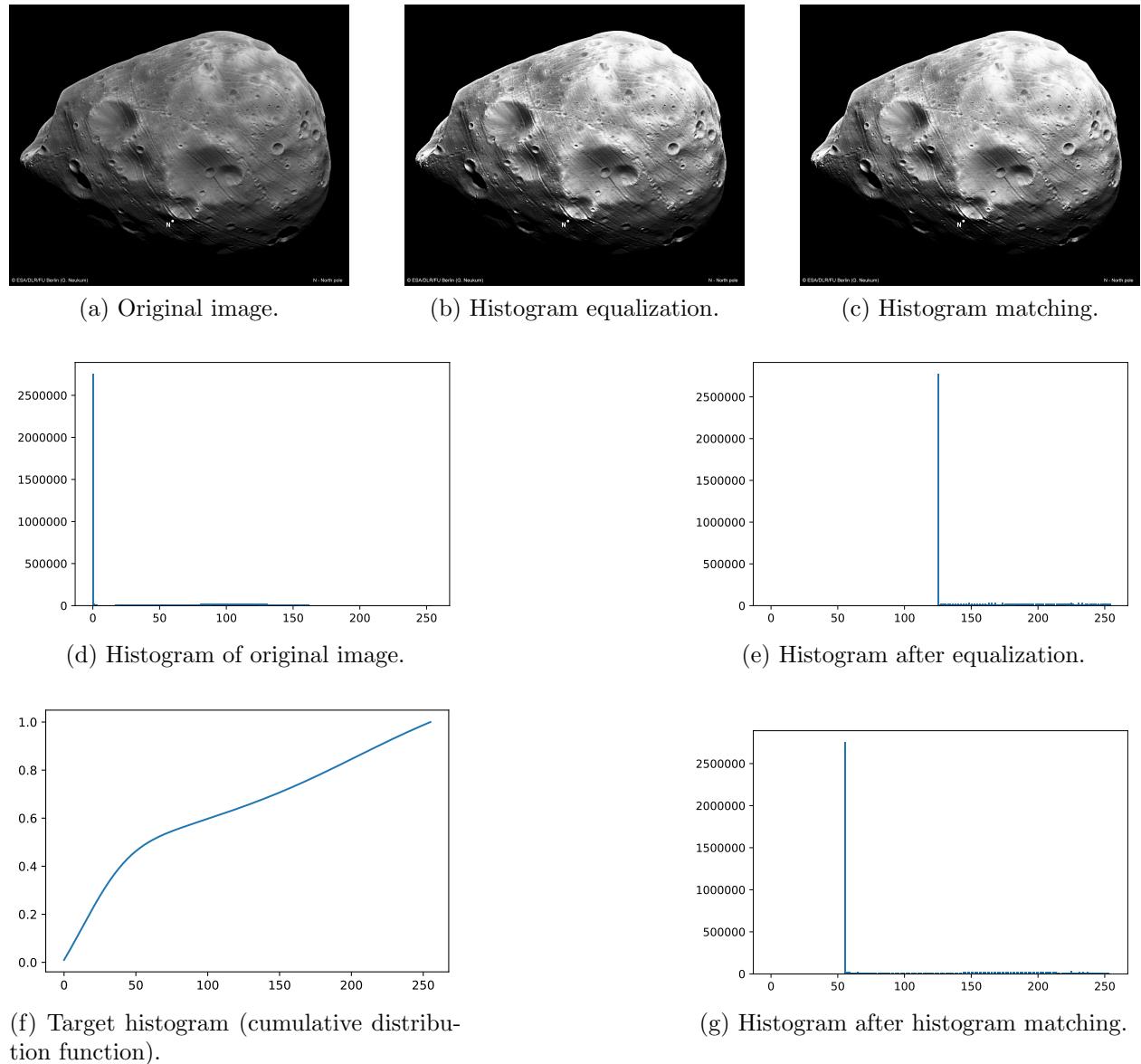


Figure 4: Histogram matching.