1 Matlab correction

1.1 Intensity transformations

At first, the image is normalized between 0 and 1.

```
A=imread('osteoblaste.tif');
A=double(A);
A=A/255; % ensure values between 0 and 1
```

1.1.1 Gamma transform

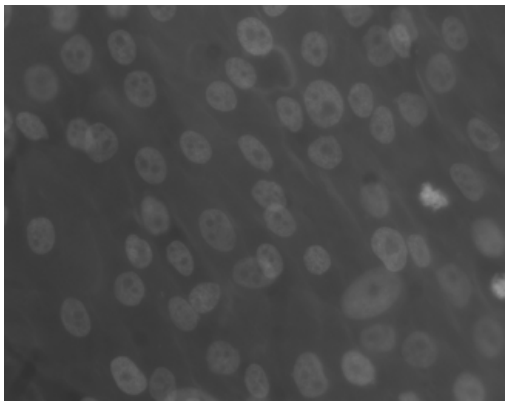
The matlab function imadjust is used to adjust gamma. Be careful that the range given in argument does not imply a strict gamma correction. Results are shown in Fig.1.

```
figure
subplot(2,2,1);imshow(A);title('original image');

Ar=imadjust(A,[min(min(A)) max(max(A))],[0 1],1);
subplot(2,2,2);imshow(Ar);title('enhanced, g=1');

Ar=imadjust(A,[0.25 0.75],[0 1],0.5);
subplot(2,2,3);imshow(Ar);title('enhanced, g=0.5');

Ar=imadjust(A,[0.25 0.5],[0 1],2);
subplot(2,2,4);imshow(Ar);title('enhanced, g=2');
```

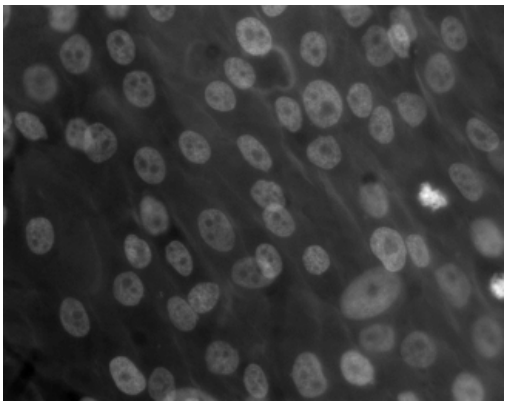
1.1.2 Contrast stretching

The function used will tend to saturate values, see Fig.2.

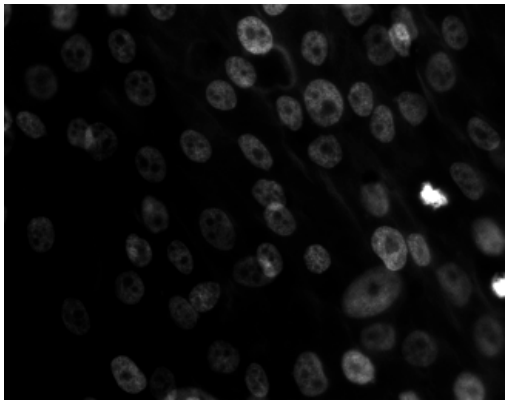
```
m=mean(mean(A));
figure
subplot(2,2,1);imshow(A);title('original image');
Ar=1./(1+(m./(A+eps)).^5);
subplot(2,2,2);imshow(Ar);title('contrast stretching : E=5');
Ar=1./(1+(m./(A+eps)).^10);
subplot(2,2,3);imshow(Ar);title('contrast stretching : E=10');
Ar=1./(1+(m./(A+eps)).^1000);
subplot(2,2,4);imshow(Ar);title('contrast stretching : E=1000');
```



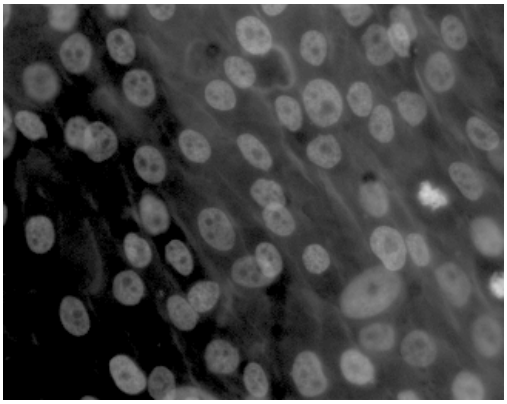
(a) Original image.



(b) $\gamma = 1$.

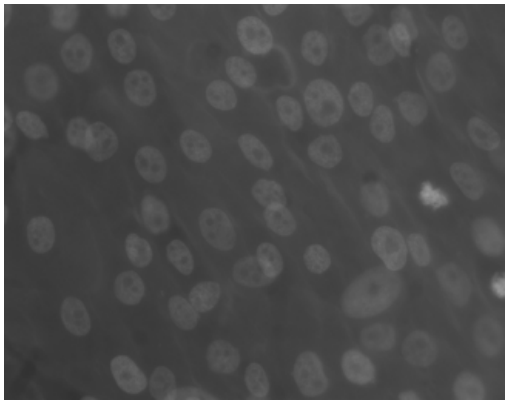


(c) $\gamma = 2$.

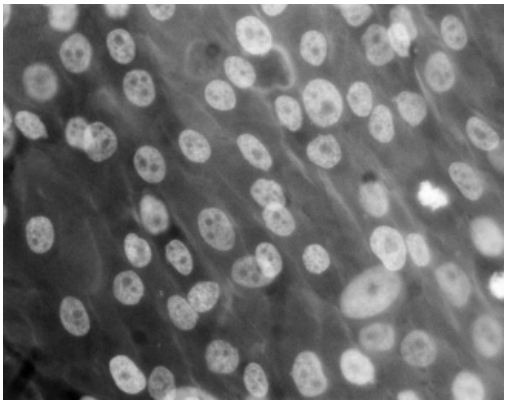


(d) $\gamma = 0.5$.

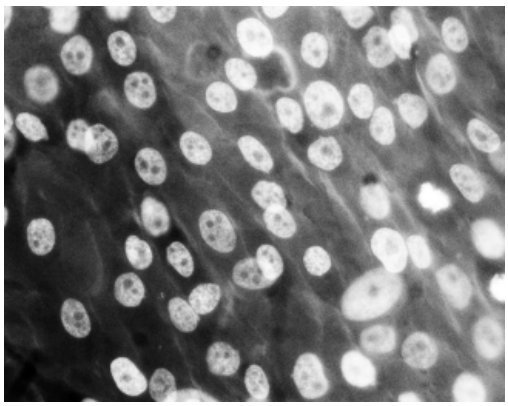
Figure 1: Gamma transform. Notice that these transforms are not exactly gamma transforms (due to the range in argument).



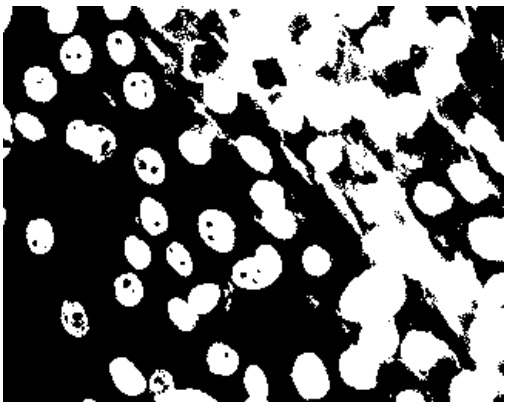
(a) Original image.



(b) $E = 5$.



(c) $E = 10$.



(d) $E = 1000$.

Figure 2: Constrast stretching.

1.2 Histogram equalization

The principle of this operation is to evaluate a lookup-table in order to perform the transformation. This LUT is the cumulative distribution function, and can be applied by using an interpolation function interp1 or the LUT function intlut. The results are shown in Fig.3.

```
A=imread('osteoblaste.tif');
Ar=histeq(A);
figure
subplot(3,2,1);imshow(A);title('original image');
subplot(3,2,2);imshow(Ar);title('histogram equalization');
subplot(3,2,3);imhist(A);title('original histogram');
subplot(3,2,4);imhist(Ar);title('equalized histogram');
hnorm = imhist(A)./numel(A);
cdf=255.*cumsum(hnorm);
subplot(3,2,5);plot(1:1:256,cdf);title('LUT (cdf)');
axis([0 255 0 255]);
```

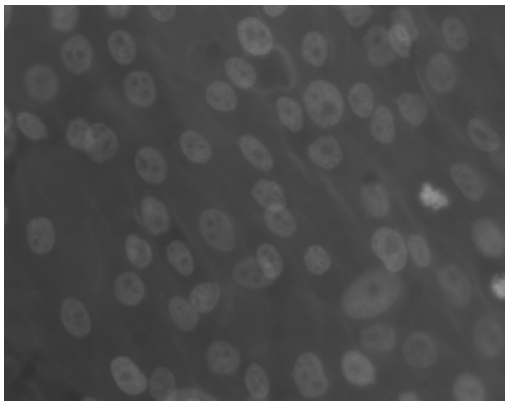
The previous code uses the matlab function for histogram equalization. You can code it as follows:

```
function I2 = histo_eq(I)
%
%    histogram equalization, version with look-up-table
%    I: original image, with values in 8 bits integer
%
[hist, edges] = histcounts(I, 0:256);
cdf = cumsum(hist);
cdf = cdf / cdf(end);

% the LUT could be applied by this function:
%I2 = intlut(I, uint8(255*cdf));

I2 = interp1(edges(1:end-1), cdf, double(I(:)));
I2 = uint8(255 * I2);
I2 = reshape(I2, size(I));
```

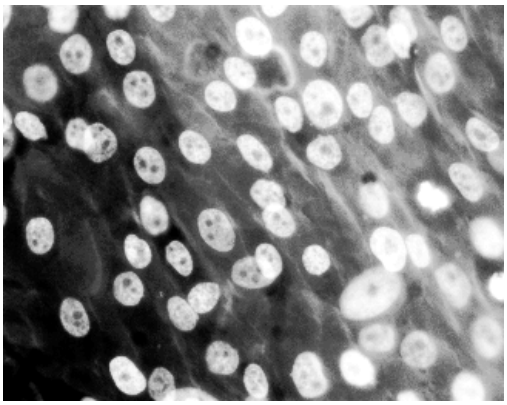
1.3 Histogram matching

The histogram matching will be applied in the Phobos image (see credits).

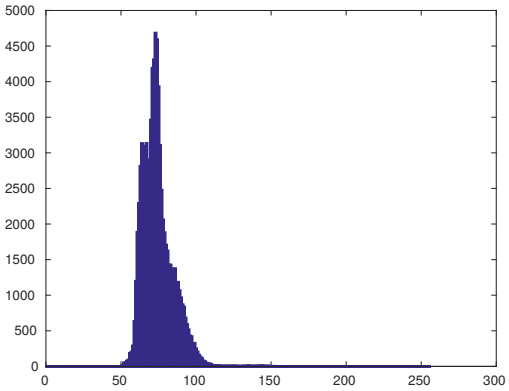
```
A=imread('phobos.jpg');
```



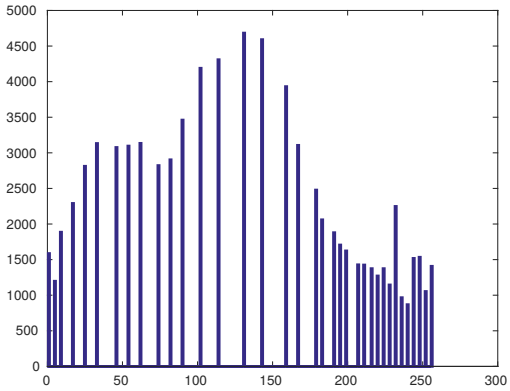
(a) Original image.



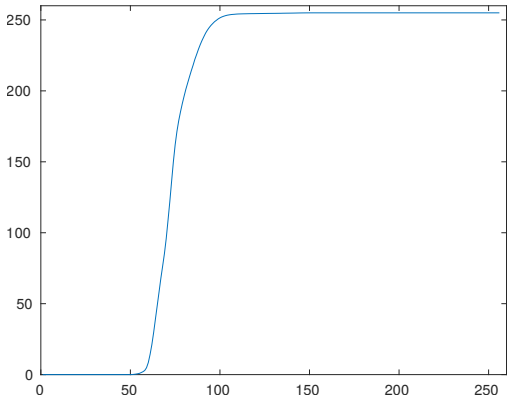
(b) Histogram equalization.



(c) Histogram of original image.



(d) Histogram of enhanced image.



(e) LUT (cumulative distribution function).

Figure 3: Histogram equalization.

First, the function to generate the probability density function is an addition of two Gaussian functions (normalized):

```
function p =twomodegauss(m1, sig1, m2, sig2, A1, A2,k)
c1 = A1*(1/((2*pi)^.5*sig1));
k1 =2*(sig1^2);
c2 = A2*(1/((2*pi)^.5*sig2));
k2 = 2*(sig2^2);
z = linspace(0,1,256);
p = k+c1*exp(-((z-m1).^ 2)./ k1)+ c2 *exp(-((z-m2).^2)./k2);
p = p./ sum(p(:));
```

Then, the histogram matching is performed as follows:

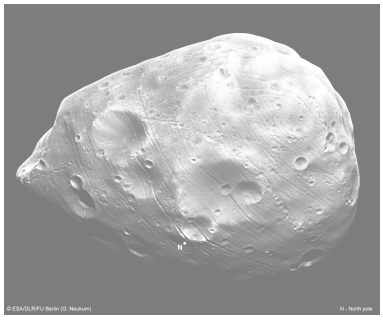
```
% histogram to match
p=twomodegauss(0.05,0.1,0.8,0.2,0.04,0.01,0.002);

% histogram matching, matlab version
Ar=uint8(histeq(A,p));
figure
subplot(3,2,6);plot(p);title('model of bi-modal histogram');
xlim([0 255])
subplot(3,2,1);viewImage(A);title('original image');
subplot(3,2,2);viewImage(Ar);title('enhanced image');
subplot(3,2,3);imhist(A,256);title('original histogram');
subplot(3,2,4);imhist(Ar,256);title('matched histogram');
```

The histeq matlab function handles histogram matching. The following code is also proposed. These are illustrated in Fig.4.

```
function I2 = histo_matching(I, cdf_target)
%
%    histogram matching, version with look-up-table
%    I: original image, with values in 8 bits integer
%
[hist, edges] = histcounts(I, 0:256);
cdf = cumsum(hist);
cdf = cdf / cdf(end);

% 1st apply histogram equalization
LUT = interp1(edges(1:end-1), cdf, double(I(:)));

% the apply inverse transformation to match target cdf
im2 = interp1(cdf_target, edges(1:end-1), LUT);

% finally, reshape and convert to uint8
```

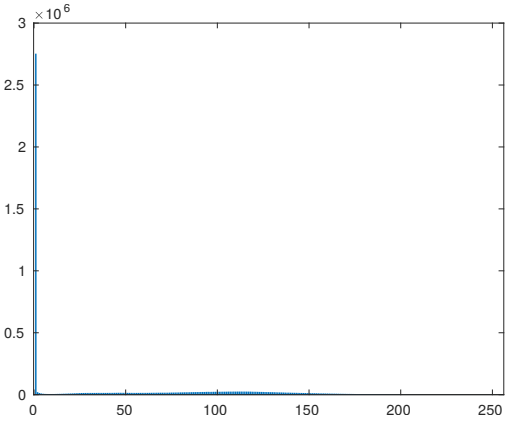
```
I2 = reshape(im2, size(I,1), size(I,2));
I2 = uint8(I2);
```

(a) Original image.

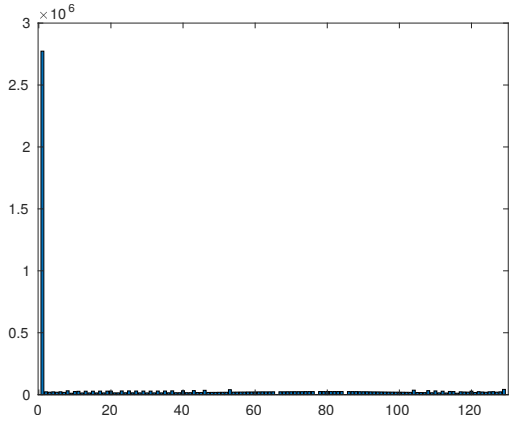


(b) Histogram equalization.

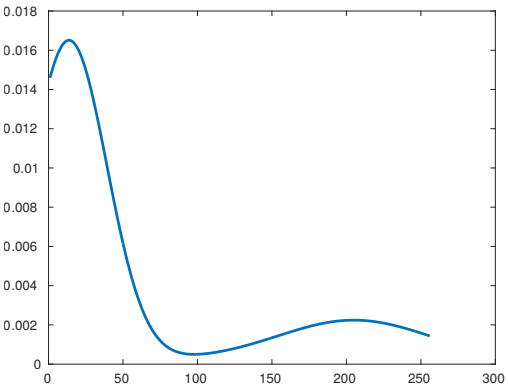
(c) Histogram matching.



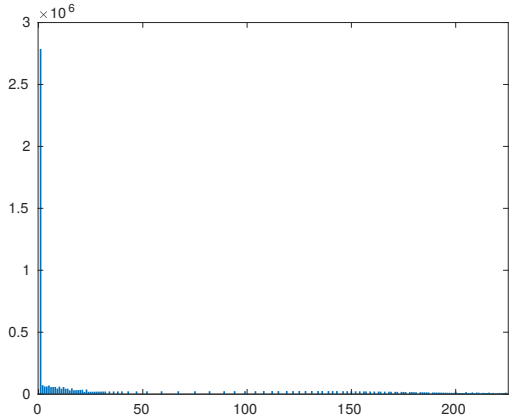
(d) Histogram of original image.



(e) Histogram after equalization.



(f) Target histogram (probability density function).



(g) Histogram after histogram matching.

Figure 4: Histogram matching.