1 Python correction

1.1 Elementary operations

The most important function to code is the graytone transformation function. It considers M as the maximum value (the absolute white). This code means that the absolute white cannot be reached, and thus $f \in [0; M[$. After discretizing the gray values, $F \in [0; M-1]$.

```python
"""
file LIP.py
LIP simple module
USAGE: import LIP

"""
import numpy as np

def graytone(F, M):
    # graytone function transform
    # M: maximal value
    # F: image function
    f = M-np.finfo(float).eps-F;
    return f;

def phi(f, M):
    # LIP isomorphism
    # f: graytone function
    # M: maximal value
    l = -M * np.log(-f/M+1);
    return l

def invphi(l, M):
    # inverse isomorphism
    f = M*(1-np.exp(-l/M));
    return f

def plusLIP(a, b, M):
    # LIP addition
    z = a+b-(a*b)/M;
    return z;

def timesLIP(alpha, x, M):
    # LIP multiplication by a real
    z = M-M*(np.ones(x.shape)-x/M)**alpha;
    return z;
```

1.2 LIP dynamic expansion

. The optimal value for dynamic expansion is given by λ_0:

$$\lambda_0(f) = \arg\max_{\lambda} \{\max(\lambda \triangle\!\!\!\!\times f) - \min(\lambda \triangle\!\!\!\!\times f)\}$$

Let $A(\lambda) = \max(\lambda \triangle f) - \min(\lambda \triangle f) = \lambda \triangle \max(f) - \lambda \triangle \min(f)$ and $B = \ln(1 - \min(f)/M)$ et $C = \ln(1 - \max(f)/M)$.

$$
\begin{aligned}
A'(\lambda) = 0 \quad &\Leftarrow \quad [(M - M\exp(\lambda C)) - (M - M\exp(\lambda B))]' = 0 \\
&\Leftarrow \quad [\exp(\lambda B) - \exp(\lambda C)]' = 0 \\
&\Leftarrow \quad B\exp(\lambda B) - C\exp(\lambda C) = 0 \\
&\Leftarrow \quad \ln(B) + \lambda B = \ln(C) - \lambda C \\
&\Leftarrow \quad \lambda = \frac{\ln(C) - \ln(B)}{B - C} \\
&\Leftarrow \quad \lambda = \frac{\ln(C/B)}{B - C}
\end{aligned}
$$

Thus, yielding to:

$$
\lambda_0(f) = \frac{\ln\left(\dfrac{\ln(1 - \max(f)/M)}{\ln(1 - \min(f)/M)}\right)}{\ln\left(\dfrac{M - \min(f)}{M - \max(f)}\right)}
$$

This is coded in python by:

```python
def computeLambda(f, M):
    # compute optimal value for dynamic expansion
    B = np.log(1-f.min()/M);
    C = np.log(1-f.max()/M);
    l = np.log(C/B)/(B-C);
    return l;
```

1.3 Complete code

This code uses a transform called histogram equalization. This version proposes our own code for the histogram equalization.

```python
import LIP
import numpy as np
from scipy import misc
import matplotlib.pyplot as plt

import skimage

""" Histogram equalization
"""
def histeq(im,nbr_bins=256):
   #get image histogram
   imhist,bins = np.histogram(im.flatten(),nbr_bins,normed=True)
   cdf = imhist.cumsum() #cumulative distribution function
   cdf = 255 * cdf / cdf[-1] #normalize

   #use linear interpolation of cdf to find new pixel values
```

```
    im2 = np.interp(im.flatten(),bins[:-1],cdf)

    return im2.reshape(im.shape), cdf
```

It is compared to the LIP dynamic enhancement. The results are illustrated in Fig. 1.

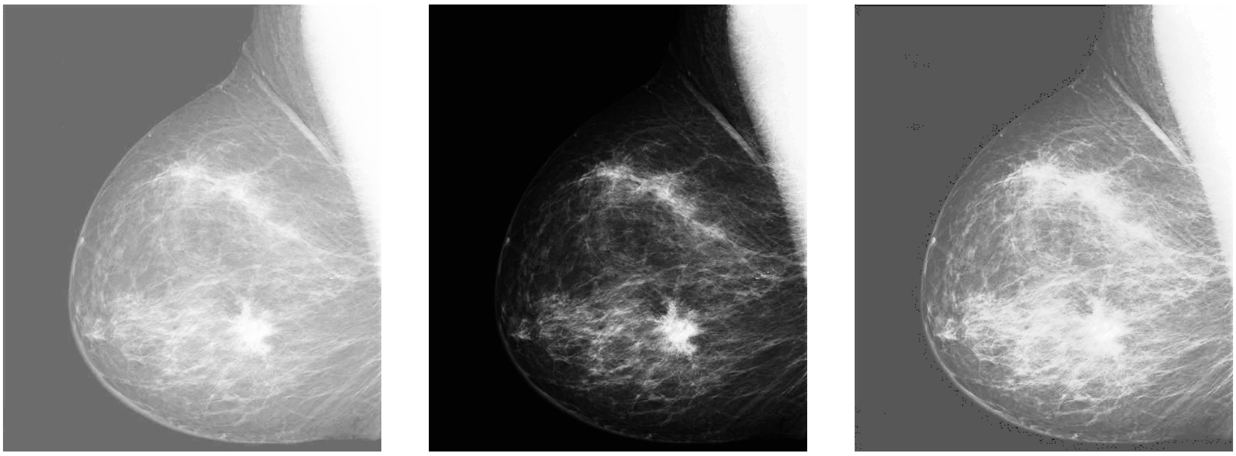
```
M = 256.

# reads original image
B = misc.imread("breast.jpg");

# conversion to gray-tones (see LIP definition)
tone = LIP.graytone(B, M);
D = LIP.graytone(LIP.timesLIP(.5, tone, M), M);

# compute optimal enhancement value
l = LIP.computeLambda(tone, M);
print "lambda: %f" % l
# apply enhancement and get back into classical space
E = LIP.graytone(LIP.timesLIP(l, tone, M), M);

# histo equalization, for comparison purposes
heq,cdf = histeq(B);

# display results
plt.figure();
plt.subplot(1,3,1); plt.imshow(E/M, cmap=plt.cm.gray, vmin=0, vmax=1); plt.
    title('dynamic expansion');
plt.subplot(1,3,2); plt.imshow(B/M, cmap=plt.cm.gray, vmin=0, vmax=1); plt.
    title('original image');
plt.subplot(1,3,3); plt.imshow(heq/M, cmap=plt.cm.gray, vmin=0, vmax=1); plt.
    title('after histo equalization');
```

1.4 Edge detection

When applying an operator in the LIP framework, it is better to apply first the isomorphisme, then the operator, and then get back into the classical space. This is applied for example for an edge detection operator.

The method applied here is the Sobel gradient. The results are presented in Fig. 2.

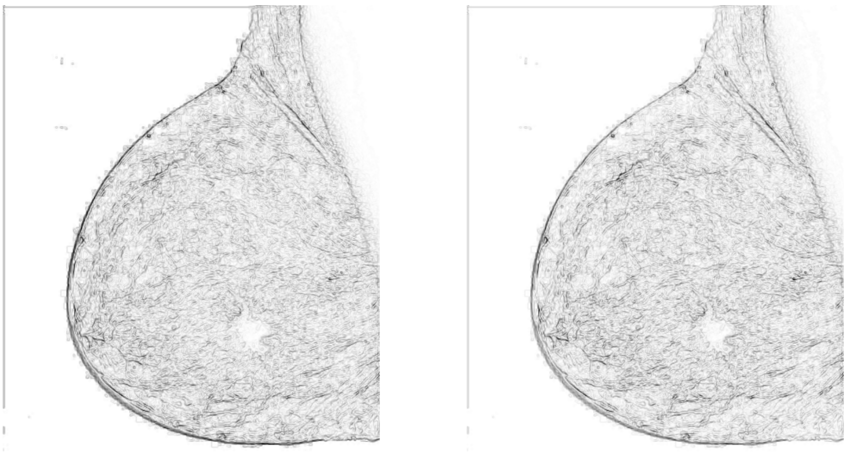
```
# go into LIP space
tonelip = LIP.phi(tone, M);

# apply Sobel filter
sobellip = skimage.filter.sobel(tonelip);
plt.figure();
plt.subplot(1,2,1);plt.imshow(sobellip, cmap=plt.cm.gray); plt.title('LIP
    Sobel edge detection')

# apply Sobel filter in the classic space
```



(a) Original image (b) LIP enhancement. (c) Histogram equalization.

Figure 1: Comparison of the LIP enhancement with the classical histogram equalization method.

```
sobel = skimage.filter.sobel(B);
plt.subplot(1,2,2); plt.imshow(sobel, cmap=plt.cm.gray); plt.title('Sobel edge detection')
```



(a) LIP Sobel edge detection. (b) Classic Sobel edge detection.

Figure 2: Sobel edge detection