1 Python correction

```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from skimage import io
from scipy import ndimage
from matplotlib import cm
```

1.1 Main function

This code is used as the main function to perform shape-from-focus reconstruction.

```python
volumes = ['cornee', 'vickers'];
for v in volumes:
    I = io.imread('volume_'+v+'.tif');
    I = I.astype('float');
    F = np.zeros(I.shape)
    N = 11;

    myfunctions = [variance, tenengrad, sml];
    for f in myfunctions:
        for i,im in enumerate(I):
            F[i] = f(im, N);

        # Evaluates altitudes and textures
        Z = np.argmax(F, axis=0);
        Z = ndimage.minimum_filter(Z, size=5);
        T = extractTexture(I, Z);
```

The extraction of the texture from the altitudes (indexes) in the stack of images is performed by the following function:

```python
def extractTexture(I, Z):
    """
    Extract texture from stack of images I, where Z is the index (altitude)
    I: stack of images, of shape (n, X, Y)
    Z: index of SFF maximum (of shape (X,Y)), values are between 0 and n-1
    returns basically I[Z(i,j), i, j)] for all (i,j)
    """
    m,n = I.shape[1:]
    ii,jj = np.ogrid[:m,:n]
    T = I[Z, ii, jj];
    return T;
```

1.2 Sum of Modified Laplacian

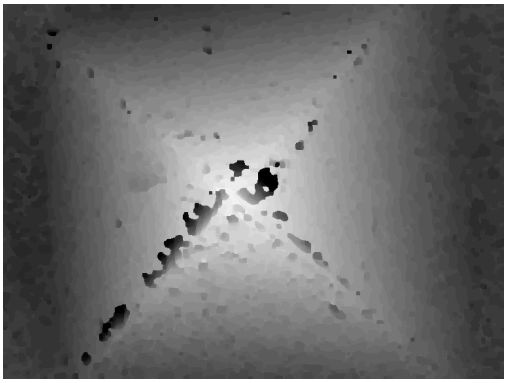
Results are illustrated in Fig.1.

```python
def sml(I, N):
    """
    SFF measure, SUM of modified Laplacian
    I: image
    N: neiborhood size
    returns: SFF measure for each pixel
    """
    h = np.array([[-1, 2, -1]]);
    ML = np.abs(ndimage.convolve(I, h)) + np.abs(ndimage.convolve(I, np.transpose(h)));
    S=ndimage.uniform_filter(ML, N);
    return S;
```

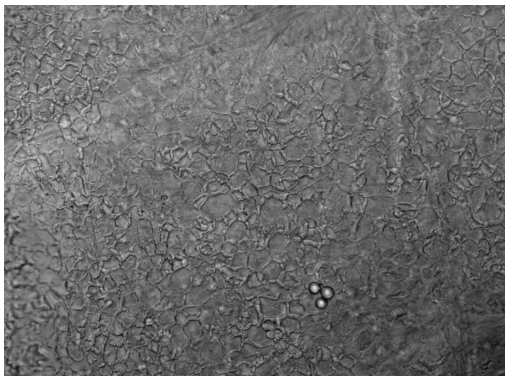
1.3 Variance

The focus measure based on the variance i a really simple method that works in most cases, see Fig.2.

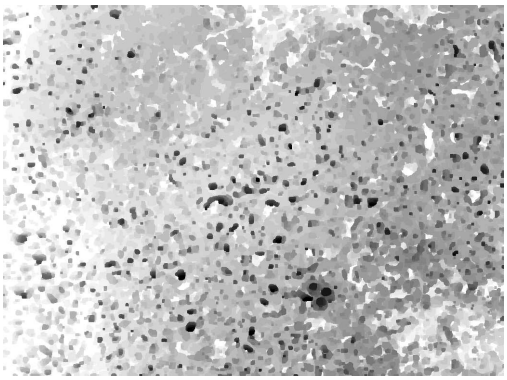
(a) Texture.



(b) Altitudes.

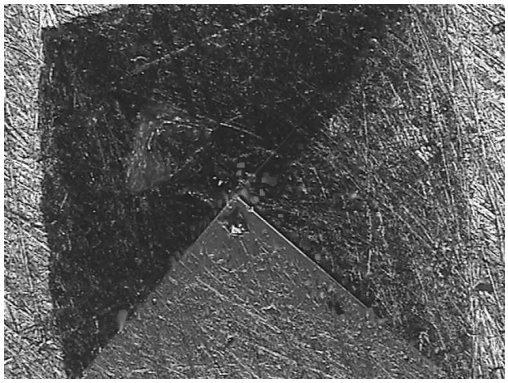


(c) Texture.

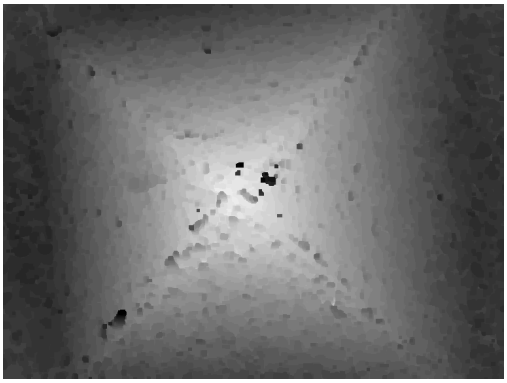


(d) Altitudes.

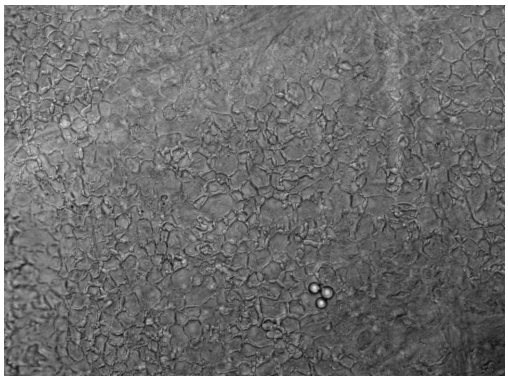
Figure 1: Texture and altitude reconstruction with the SML method.



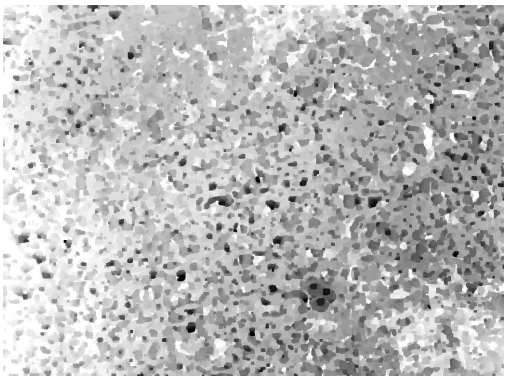
(a) Texture.



(b) Altitudes.



(c) Texture.



(d) Altitudes.

Figure 2: Texture and altitude reconstruction with the variance method.

```
def variance(I, N):
    """
    SFF measure
    I: image
    N: neighborhood size
    returns: SFF measure for each pixel, results is the same shape as I
    """
    M = ndimage.uniform_filter(I, N);
    D2 = (I - M) ** 2;
    V = ndimage.uniform_filter(D2, N);
    return V;
```

1.4 Tenengrad

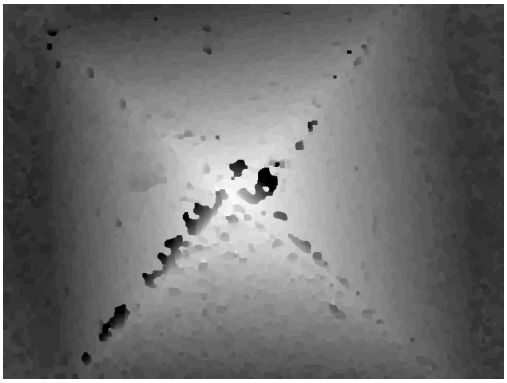
The tenengrad method is base on a Sobel filter, see Fig.3.

```
def tenengrad(I, N):
    """
    SFF measure, Tenengrad method
    I: image
    N: neighborhood size
    returns: SFF measure for each pixel, results is the same shape as I
    """
    Sx = ndimage.sobel(I, axis=0);
    Sy = ndimage.sobel(I, axis=1);
    S = np.hypot(Sx, Sy);
    T = ndimage.uniform_filter(S, N);
    return T;
```

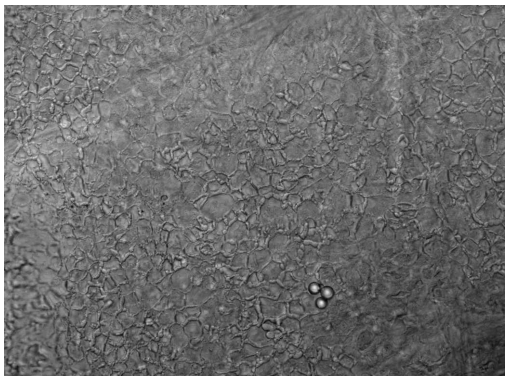
1.5 Variance of Tenengrad

The variance of Tenengrad is an improvement of the Tenengrad method, see Fig.4.

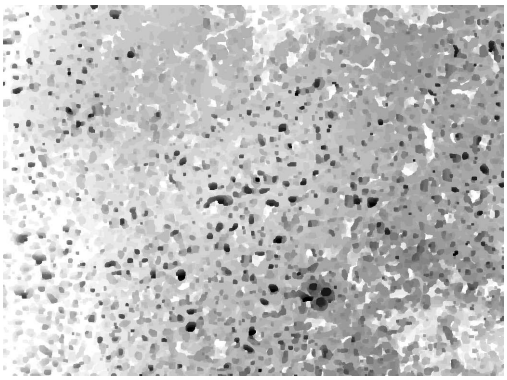
(a) Texture.



(b) Altitudes.



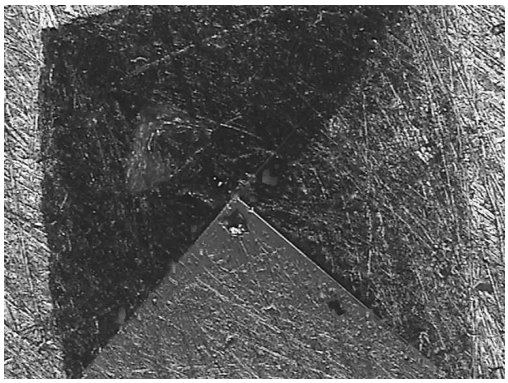
(c) Texture.



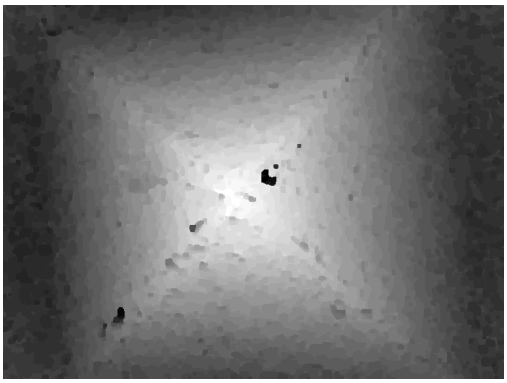
(d) Altitudes.

Figure 3: Texture and altitude reconstruction with the SML method.

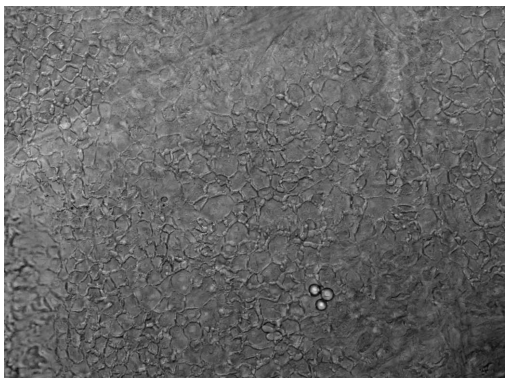
```python
def varianceTenengrad(I, N):
    """
    SFF measure, variance of Tenengrad
    I: image
    N: neiborhood size
    returns: SFF measure for each pixel
    """
    Sx = ndimage.sobel(I, axis=0);
    Sy = ndimage.sobel(I, axis=1);
    S = np.hypot(Sx, Sy);
    vt = variance(S, N);
    return vt;
```



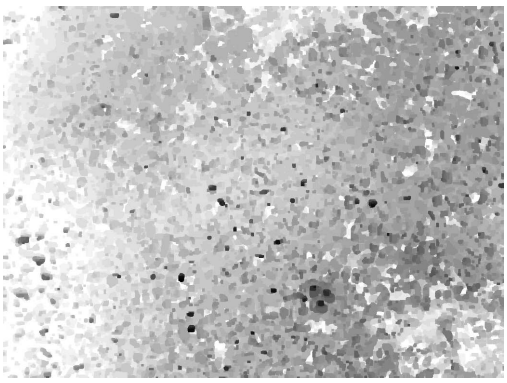
(a) Texture.



(b) Altitudes.



(c) Texture.



(d) Altitudes.

Figure 4: Texture and altitude reconstruction with the variance of Tenengrad method.