1 Python correction

he following imports are used.

```
import numpy as np
from scipy import misc
import matplotlib.pyplot as plt
import glob
import seaborn as sn
import pandas as pd
import os
from sklearn.cluster import KMeans
```

1.1 LBP computation

Each pixel is given a specific 8 bits value according to a code as follows.

```
def LBP(I):
    B = np.zeros(np.shape(I));
    code = np.array([[1,2,4],[8,0,16],[32,64,128]]);

    # loop over all pixels except border pixels
    for i in np.arange(1,I.shape[0]-2):
        for j in np.arange(1, I.shape[1]-2):
            w = I[i-1:i+2, j-1:j+2];
            w = w >= I[i,j];
            w = w * code;
            B[i,j] = np.sum(w);
```

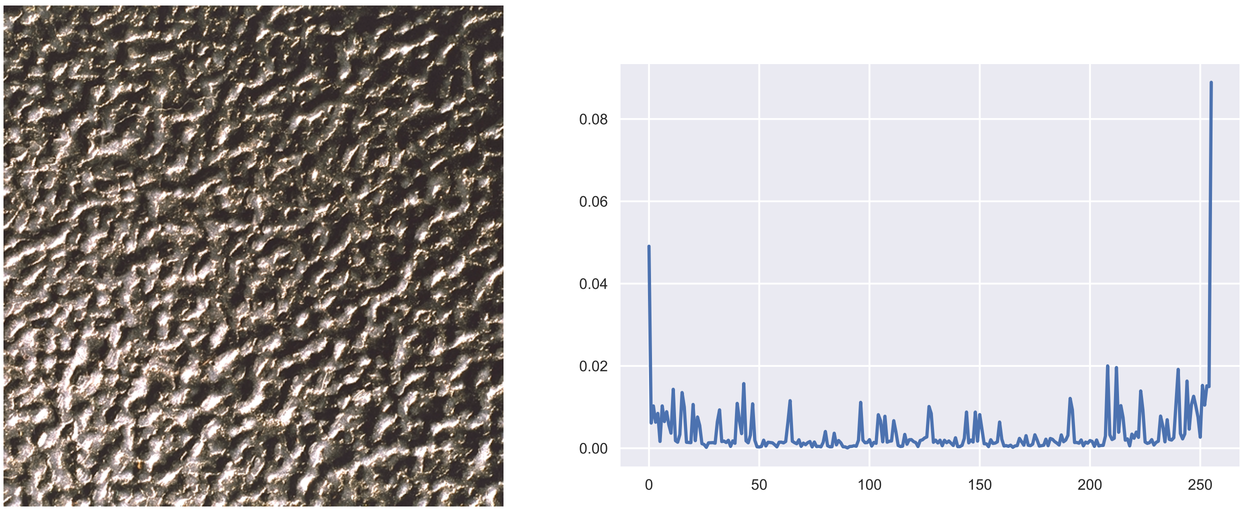
Then, all values (except for border values) are summarized in the histogram.

```
h,edges = np.histogram(B[1:-1, 1:-1], density=True, bins=256);
```

For the first image of sand, the histogram is shown in Fig.1.

1.2 Classification

For all images of the same family, the LBP are computed and represented in the same graph. The histograms really look similar (see Fig.2). The following code is used for the "sand" family.



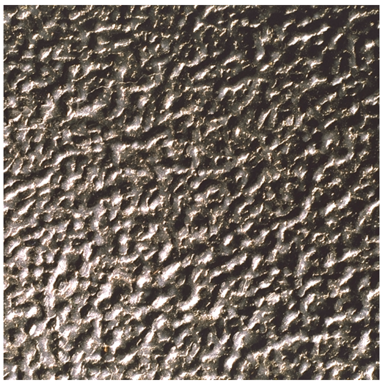
(a) Texture image. (b) LBP of texture.

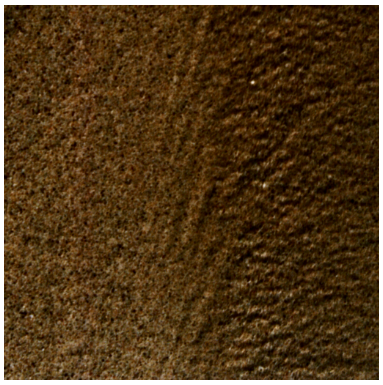
Figure 1: Illustration of the Local Binary Pattern computed on an entire image.

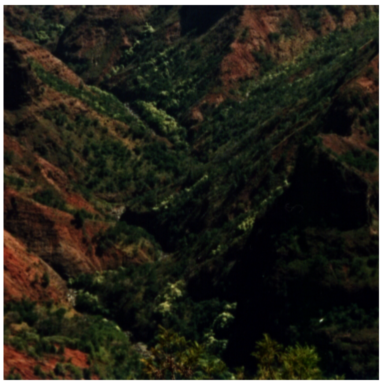
```python
classes = ['Terrain', 'Metal', 'Sand'];
names = [];
hh = [];

for c in classes:
    print(c);
    fig=plt.figure();
    for file in sorted(glob.glob('../matlab/images/'+ c + '*.bmp')):
        names.append(os.path.basename(file));
        I = misc.imread(file);
        I = I[:,:,1];
        h, edges = LBP(I);
        plt.plot(h);
        hh.append(h);
```

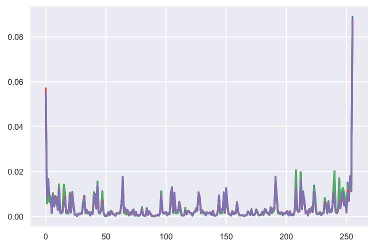
A distance criterion is used to compare the different histograms: the classical SAD (Sum of Absolute Differences) gives a numerical values. All pairs of distances are concatenated in a matrix, displayed as an image in Fig.3.

```python
# compute distance between LBPs
n = len(hh);
dists = np.zeros((n, n))
for i in np.arange(n):
    for j in np.arange(n):
        dists[i,j] = np.sum(np.abs(hh[i]-hh[j]))

fig=plot_dists(dists, names);
```

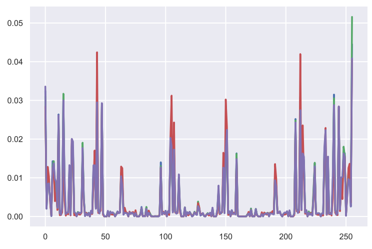

(a) Metal image example.


(b) Sand image example.

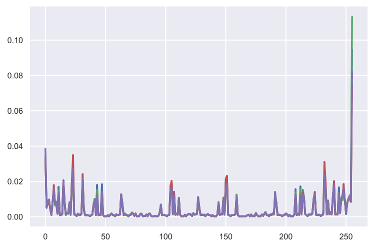

(c) Terrain image example.



(d) Four metal images.



(e) Four sand images.

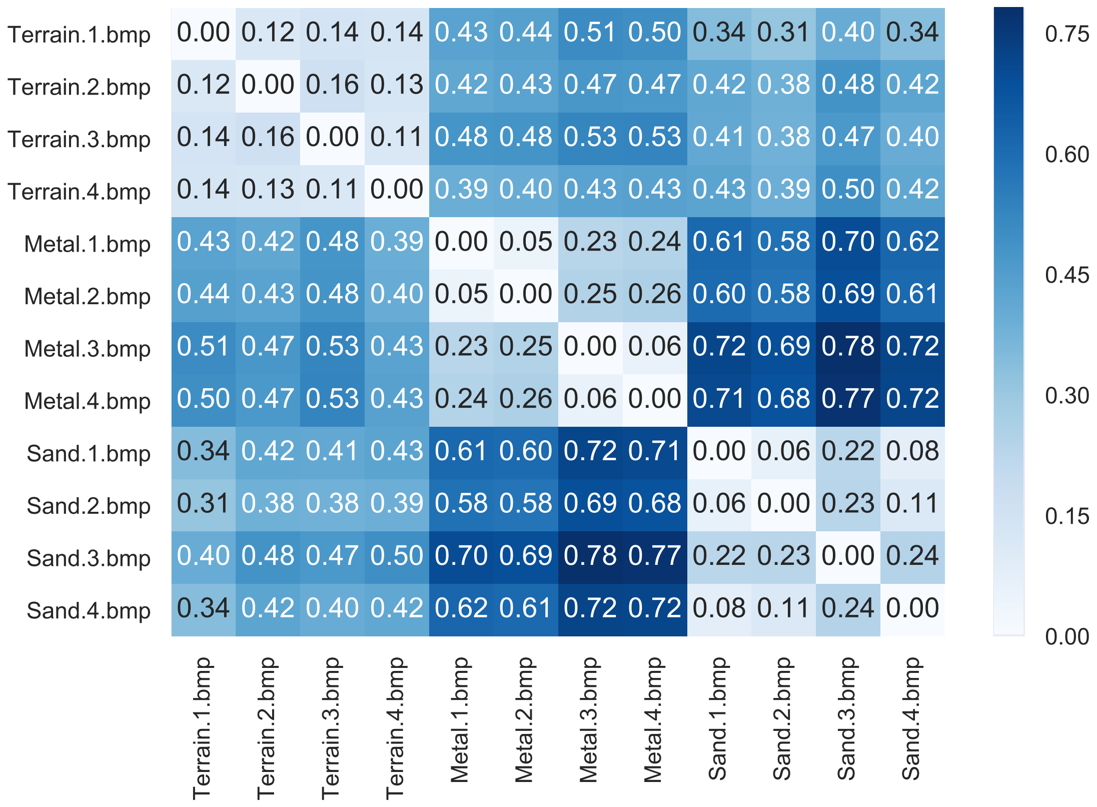


(f) Four terrain images.

Figure 2: Illustration of the LBP of 4 images of each family. The histogram are almost equivalent, which shows that this descriptor can be employed to discriminate between the different families.

In order to display this matrix, the module seaborn is used.

```python
def plot_dists(dists, classes, cmap=plt.cm.Blues):
    """
    Plot matrix of distances
    dists: all computed distances
    classes: labels to be used
    cmap: colormap

    returns: figure that can be used for pdf export
    """
    df_cm = pd.DataFrame(dists, index = classes, columns = classes);

    fig = plt.figure();
    sn.set(font_scale=.8)
    sn.heatmap(df_cm, annot=True, cmap = cmap, fmt='.2f');

    return fig;
```



	Terrain.1.bmp	Terrain.2.bmp	Terrain.3.bmp	Terrain.4.bmp	Metal.1.bmp	Metal.2.bmp	Metal.3.bmp	Metal.4.bmp	Sand.1.bmp	Sand.2.bmp	Sand.3.bmp	Sand.4.bmp
Terrain.1.bmp	0.00	0.12	0.14	0.14	0.43	0.44	0.51	0.50	0.34	0.31	0.40	0.34
Terrain.2.bmp	0.12	0.00	0.16	0.13	0.42	0.43	0.47	0.47	0.42	0.38	0.48	0.42
Terrain.3.bmp	0.14	0.16	0.00	0.11	0.48	0.48	0.53	0.53	0.41	0.38	0.47	0.40
Terrain.4.bmp	0.14	0.13	0.11	0.00	0.39	0.40	0.43	0.43	0.43	0.39	0.50	0.42
Metal.1.bmp	0.43	0.42	0.48	0.39	0.00	0.05	0.23	0.24	0.61	0.58	0.70	0.62
Metal.2.bmp	0.44	0.43	0.48	0.40	0.05	0.00	0.25	0.26	0.60	0.58	0.69	0.61
Metal.3.bmp	0.51	0.47	0.53	0.43	0.23	0.25	0.00	0.06	0.72	0.69	0.78	0.72
Metal.4.bmp	0.50	0.47	0.53	0.43	0.24	0.26	0.06	0.00	0.71	0.68	0.77	0.72
Sand.1.bmp	0.34	0.42	0.41	0.43	0.61	0.60	0.72	0.71	0.00	0.06	0.22	0.08
Sand.2.bmp	0.31	0.38	0.38	0.39	0.58	0.58	0.69	0.68	0.06	0.00	0.23	0.11
Sand.3.bmp	0.40	0.48	0.47	0.50	0.70	0.69	0.78	0.77	0.22	0.23	0.00	0.24
Sand.4.bmp	0.34	0.42	0.40	0.42	0.62	0.61	0.72	0.72	0.08	0.11	0.24	0.00

Figure 3: Sum of Absolute Differences between the different LBP histograms of each image. 3 families of 4 textures are represented here, terrain images are in the first part, metal images in the second and sand images in the last. Black represents 0 distance and white is 1 (highest distance, the values are normalized).

The kmeans algorithm uses such a distance, and we can verify that the clustering process works as expected. The result is presented in the next box.

```python
# kmeans clustering
n=3;
k_means = KMeans(init='k-means++', n_clusters=n, n_init=10)
k_means.fit(hh);
print(k_means.labels_)
```

The result show that the kmeans algorithm perfectly performs the classification.

```
[1 1 1 1 0 0 0 0 2 2 2 2]
```