Université Abdelmalek Essâadi de Tanger
Faculté des Sciences et Techniques de Tanger
Département Génie Informatique

Content based image retrieval

Texture wavelet

Filière : Mobiquité et Big Data

Réaliser par :

AHARMOUCH Mohamed Hamza

Année universitaire : 2020/2021

Table des matières

Objectif :

Mettre au point une application Web, avec Flask comme serveur, qui permet d'implémenter les fonctionnalités de base d'un système d'indexation et de recherche d'images par le contenu. En utilisant la caractéristique suivante : Texture Wavelet

Outils :

- Flask
- React JS
- Vs Code
- Biliothèque PyWavelets

I. Texture : Définition

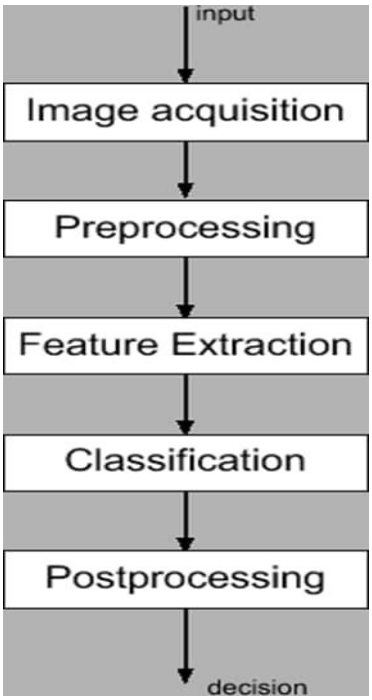
Dans le traitement d'image, l'analyse de texture consiste à calculer une série de mesures dans le but de définir une texture perçue sur une image. L'analyse de texture renvoie des informations sur l'arrangement spatial des couleurs ou des intensités dans tout ou partie de cette image.

Les données de texture d'une image peuvent être artificiellement créées (textures artificielles) ou résulter de l'analyse d'images filmées à partir de scènes ou d'objets réels (textures naturelles). L'analyse (ou caractérisation) de texture joue un rôle important dans la segmentation d'image ou dans sa classification. Les éléments apportant le plus de précisions dans la segmentation sont les fréquences spatiales et la moyenne du niveau de gris.

II. Analyse de texture et ses applications :

Les principaux objectifs de la recherche sur les textures en vision par ordinateur sont de comprendre, modéliser et traiter la texture, et finalement de simuler le processus d'apprentissage visuel humain à l'aide de technologies informatiques.

Un système de vision par ordinateur typique peut être divisé en composants tels que ceux illustrés à la figure. L'analyse de texture peut être appliquée à différentes étapes du processus. Au stade du prétraitement, les images pourraient être segmentées en régions contiguës en fonction des propriétés de texture de chaque région ; Aux étapes d'extraction de caractéristiques et de classification, les caractéristiques de texture pourraient fournir des indices pour classer des motifs ou identifier des objets.



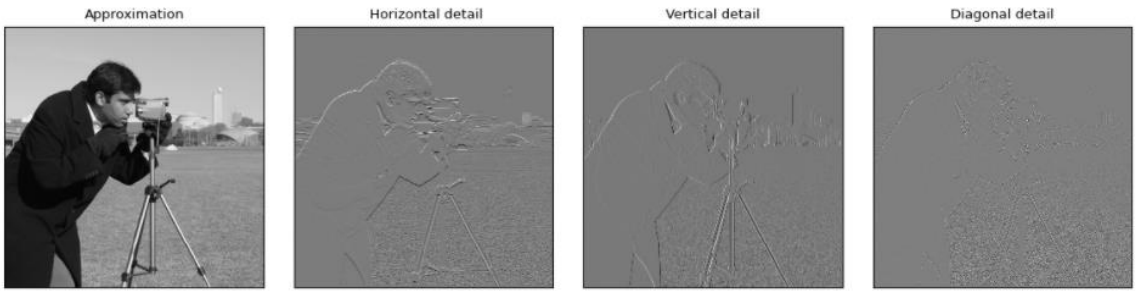
III. Explication de la bibliothèque PyWavelets :

PyWavelets est un logiciel de transformation d'ondelettes open source pour Python . Il combine une interface simple de haut niveau avec des performances de bas niveau C et Cython.

PyWavelets est très facile à utiliser et à utiliser. Installez simplement le package, ouvrez le shell interactif Python et tapez :

Voici un exemple un peu plus complexe d'application d'une transformation en ondelettes numérique à une image :

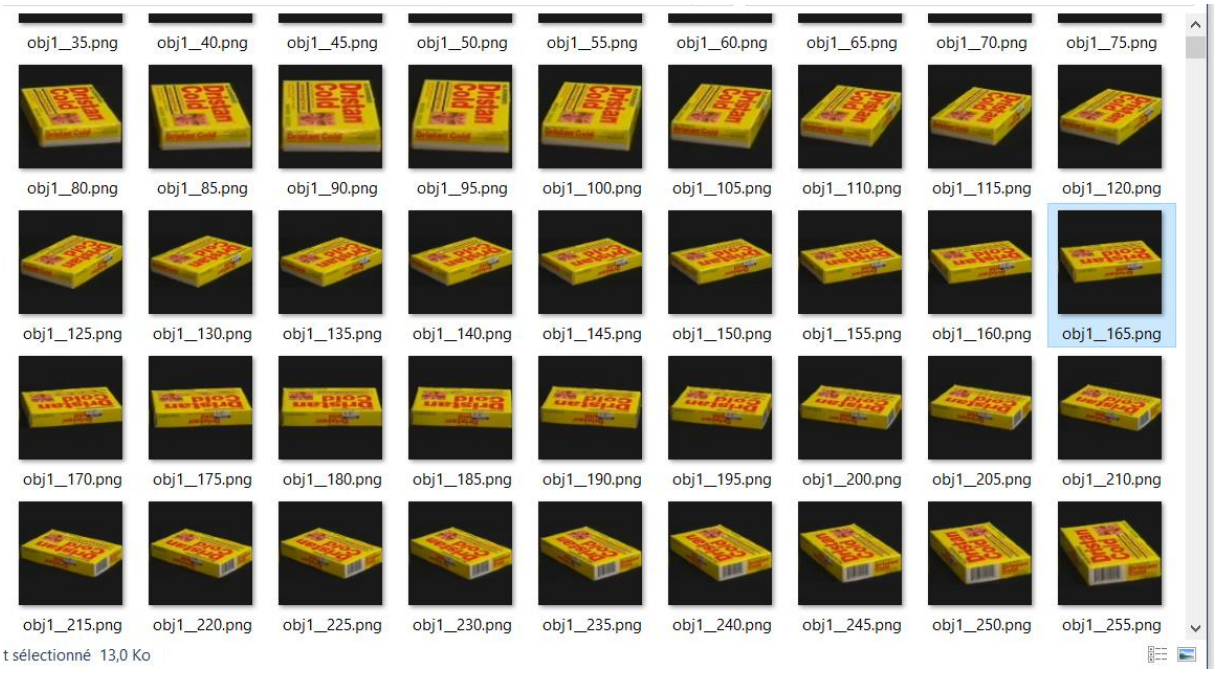
```python
import numpy as np
import matplotlib.pyplot as plt

import pywt
import pywt.data


# Load image
original = pywt.data.camera()

# Wavelet transform of image, and plot approximation and details
titles = ['Approximation', ' Horizontal detail',
          'Vertical detail', 'Diagonal detail']
coeffs2 = pywt.dwt2(original, 'bior1.3')
LL, (LH, HL, HH) = coeffs2
fig = plt.figure(figsize=(12, 3))
for i, a in enumerate([LL, LH, HL, HH]):
    ax = fig.add_subplot(1, 4, i + 1)
    ax.imshow(a, interpolation="nearest", cmap=plt.cm.gray)
    ax.set_title(titles[i], fontsize=10)
    ax.set_xticks([])
    ax.set_yticks([])

fig.tight_layout()
plt.show()
```



IV. La base de données utilisé :

COIL-100 a été collecté par le Centre de recherche sur les systèmes intelligents du Département d'informatique de l'Université de Columbia. La base de données contient des images en couleur de 100 objets. Les objets ont été placés sur un plateau tournant motorisé sur un fond noir et les images ont été prises à des poses internes de 5 degrés. Cet ensemble de données a été utilisé dans un système de reconnaissance de 100 objets en temps réel grâce auquel un capteur système pourrait identifier l'objet et afficher sa pose angulaire.

Il y a 7 200 images de 100 objets. Chaque objet a été tourné sur une plaque tournante à 360 degrés pour faire varier la pose de l'objet par rapport à une caméra couleur fixe. Les images des objets ont été prises à des intervalles de pose de 5 degrés. Cela correspond à 72 poses par objet. Là, les images ont ensuite été normalisées en taille. Les objets ont une grande variété de caractéristiques géométriques et de réflectance complexes.



V. Quelques parties du code source de l'application

On s'est basé sur l'utilisaation de la bibliotheque PyWavelets dans le traitement d'image et la décomposition de ses dérnier selon la fonction du wavelets.

PyWavelets est une bibliothèque de transformation d'ondelettes open source pour Python. Il combine une interface simple de haut niveau avec des performances de bas niveau C et Cython.

```
57      img=mpimg.imread(filepath)
58      #Multiple level DWT
59      coeffs2=pywt.wavedec2(img,'db5',mode='periodization',level=2)
60      #Get coefficients
61      cA2=coeffs2[0]
62      (cH1,cV1,cD1)=coeffs2[-1]
63      (cH2,cV2,cD2)=coeffs2[-2]
64      coefs_list=[cH1,cV1,cD1,cA2,cH2,cV2,cD2]
65      for coef in coefs_list:
66          variance = ndimage.variance(coef)
67          var_list.append(variance)
68      for coef in coefs_list:
69          labels,labels_num=ndimage.label(coef)
70          mean=ndimage.mean(coef,labels)
71          var_list.append(mean)
```

On commence par la lecture des images en utilisant la fonction imread de matplotlib.image,

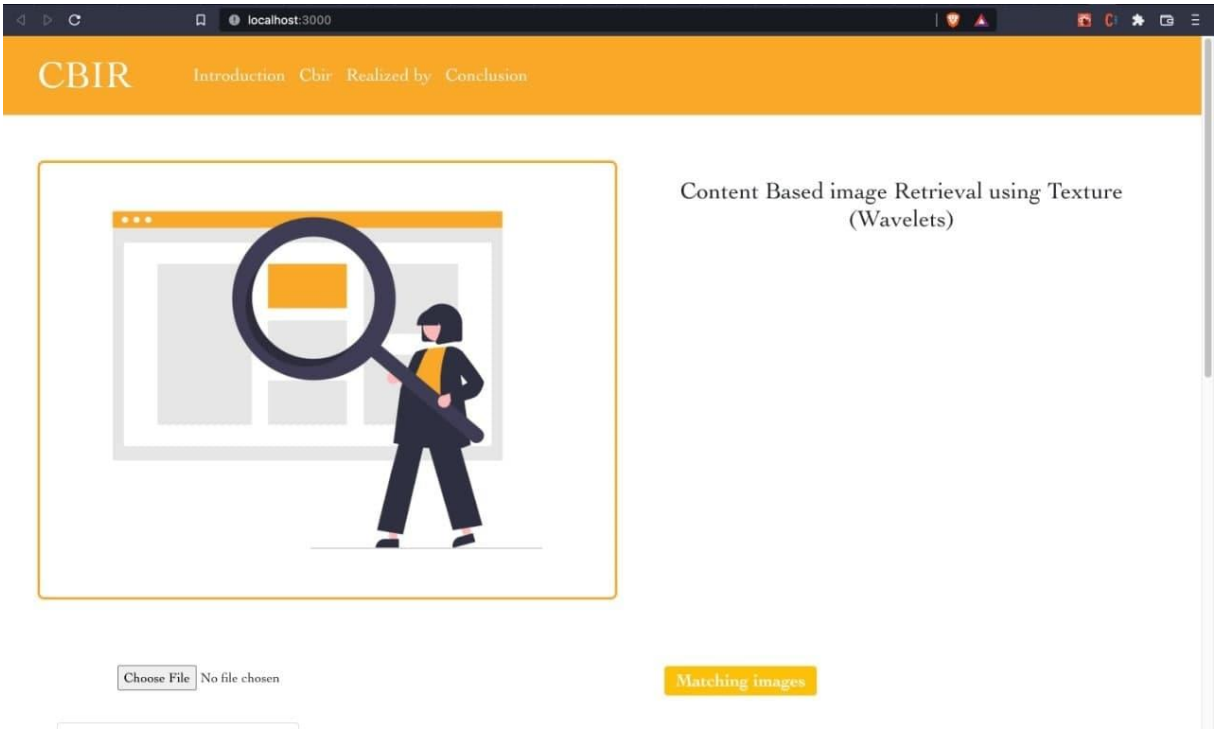
Puis on utilise pywt.wavedec2 pour calculer la transformation wavelet de l'image avec le mode periodization pour qu'il nous retourne un vecteur de coefficient de moitie de longueur de plus on précis le niveau de décomposition avec le niveau 2. Chaque niveau de décomposition retourne 4 images ['Approximation', ' Horizontal detail','Vertical detail', 'Diagonal detail'].

Donc on totale on aura 7 vecteur comme résultats final : cH1,cV1,cD1,cA2,cH2,cV2,cD2

```
72      #convert json file to dict
73      with open('var_json.json') as vj:
74          distances={}
75          list_vj = json.load(vj)
76          for d_vj in list_vj:
77              eucl_distance=eucl_dist(var_list,d_vj["var_list"])
78              path_vj=d_vj["path"]
79              distances[path_vj]=eucl_distance
80          #print(distances)
81          k=12
82          mindist4 = dict(sorted(distances.items(), key = itemgetter(1))[:k])
83          #print(mindist4)
84          filenames_1=list(mindist4.keys())
85          string = 'coil-100/'
86          filenames = list(map(lambda orig_string: string+orig_string, filenames_1))
87
88      #return f"The list is : {var_list}"
89      return filenames
90
```

En seconde lieu on calcule la distance euclidienne en utilisant la fonction eucl_dist et après on stock les résultats qui est en forme de vecteur de 14 éléments pour chaque image 7 variance et 7 moyennes dans un fichier json.

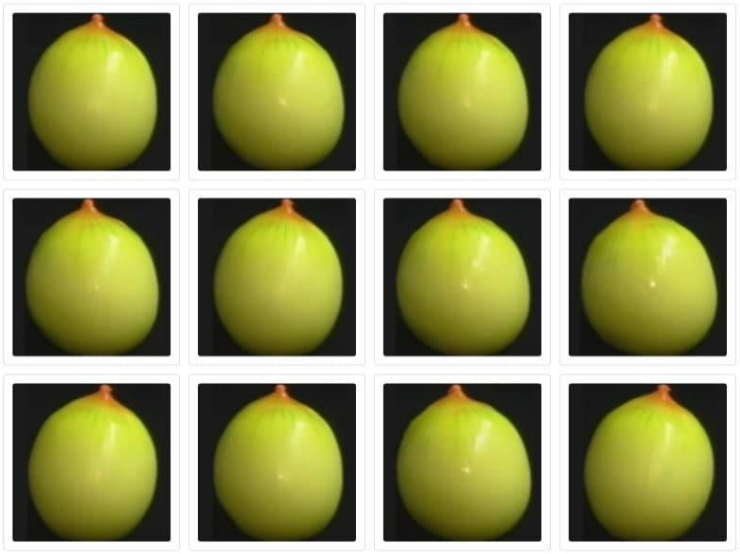
VI. Quelques captures écran de la partie frontend



Choose File obj2__80.png

Start Searching

Matching images



Choose File obj4__70.png

Start Searching

Matching images

Choose File obj99__135.png

Start Searching

Matching images