TP2 - Données Multimedia

Campedel - Juillet 2014
CES Data Scientist

Objectifs
prendre conscience du rôle du “Data Scientist” : choix des attributs, choix de la mesure de similarité

Ressources
https://docs.python.org/2/tutorial/
http://opencv-python-tutroals.readthedocs.org/
Cliquer ici pour en savoir plus sur les histogrammes dans OpenCV

Exercice 0 : Image couleur

Observez les 3 canaux de l’image

```
#et en couleur ?
image = cv2.imread(filename)
rows, col, nbChannels = image.shape
r,g,b = cv2.split(image)

plt.figure()
plt.subplot(2,2,1)
plt.imshow(image)
plt.title('image en couleur')
plt.subplot(2,2,2)
plt.imshow(b, cmap='gray')
plt.title('canal bleu')
plt.subplot(2,2,3)
plt.imshow(g, cmap='gray')
plt.title('canal vert')
plt.subplot(2,2,4)
plt.imshow(r, cmap='gray')
plt.title('canal rouge')
```

Exercice 1 : Espace de couleur
La première signature que nous avons exploitée (moyenne et variance des pixels en niveaux de gris) est rudimentaire. Bien souvent il est utile d’exploiter l’ensemble de l’information de couleur.

- ouvrir une image en couleur
- récupérer sa taille avec la propriété shape() : vous observez une 3ème valeur, qui indexe les canaux B, G et R.
- calculer un histogramme (en exploitant la fonction de OpenCV) et visualiser `cv2.calcHist`
- Bien souvent nous faisons appel à l'espace dit HSV : convertissez l'image en HSV. `cv2.cvtColor`
- calculer un nouvel histogramme et visualiser
- Commenter en exploitant deux images : leurs histogrammes BGR et HSV vous apparaissent-ils comme similaires ? différents ? Comment pourrions-nous les comparer de façon automatique ?

Exercice 2 : système de recherche par image similaire

- Recopier le script proposé : que fait ce script ?
- Exploiter différents jeux d'attributs, modifiez les images requêtes : qu'observez-vous ?
- Exploitez maintenant, pour un même jeu d'attributs, différentes façons de calculer les distances (L2, L1, Mahalanobis par exemple). Qu'observez-vous ?
- Qu'avez-vous appris ?

Indice :
Il est normal de se sentir dépassé par tous les choix possibles.

Script proposé :

```
import cv2
import matplotlib.pyplot as plt
import numpy as np

def compareFeatures( sign1, sign2 ):
	#vérification des dimensions
	L1 = len(sign1)
	L2 = len(sign2)
	flag = 1;
	if (L1 == L2 ):
		#calcul de la distance ou de la similarité
		if flag==1:
			d = 0;
			for i in range(L1):
				d=d+ (sign1[i]-sign2[i])**2
			d = np.sqrt( d )
		elif flag==2:
			d = 0;
			for i in range(L1):
				d=d+np.abs(sign1[i]-sign2[i])
		else:
			d = 0;
			for i in range(L1):
			d=d+sign1[i]/sign2[i]*np.log(sign1[i]/sign2[i])
```

```
                return d
        else:
                return np.inf

def computeFea( imageName ):
        flag = 1;
        if flag==1:
                im=cv2.imread(imageName, cv2.CV_LOAD_IMAGE_GRAYSCALE) #en gris
                mi = np.mean(im)
                vi = np.var(im)
                sign = [mi, vi]
        elif flag==2:
                im=cv2.imread(imageName)
                hsv=cv2.cvtColor(im,cv2.COLOR_BGR2HSV)
                h,s,v=cv2.split(hsv)
                mi = np.mean(h)
                vi = np.var(h)
                sign = [mi]
        elif flag==3:
                #SIFT
                im=cv2.imread(imageName, cv2.CV_LOAD_IMAGE_GRAYSCALE) #en gris
                #à implémenter !
                #
        else:
                im=cv2.imread(imageName)
                hsv=cv2.cvtColor(im,cv2.COLOR_BGR2HSV)
                sign = cv2.calcHist([hsv],[0],None,[180],[0,180])
        return sign

#A adapter à votre Path !
myImagePath = '/Users/Marine/Documents/Telecom/CESDataScientist/images/'

bComputeFeatures = 1;
#il faudrait implémenter les autres cas ;
#si on doit stocker les #attributs : ce n'est pas fait !

#la liste des images se trouve dans le répertoire indiqué par #myImagePath
listName = myImagePath + 'list.txt'

allFea = [[]]
allNames = [[]]
if bComputeFeatures:
        #Compute the signatures
        f = open(listName, 'rt')
        n = 0
        for line in f:
                w = line.split()[0]
                filename = myImagePath + w
                print(filename)
                fea = computeFea( filename )
                #save the result
                allFea.append(fea)
                allNames.append(w)
                n = n+1
```

```
        nbImages = n;
        #normalization
        #what could you imagine?
        #Is it necessary?

        f.close();
        #save features
else:
        #load features
        nbImages = len(allNames)

#Compare the request image to all stored images
#N'hésitez pas à changer d'image requête

imgReq = myImagePath + '00000001.jpeg'
feaReq = computeFea( imgReq );

f = open(listName, 'rt')
allDist = []
for n in range(nbImages):
        d = compareFeatures(allFea[n],feaReq)
        allDist.append(d)

#sort the distances
ind = np.argsort(allDist)
#Get the nbFisrt results
nbFirst = 5
plt.figure();
plt.subplot(2,3,0) #pareil que plt.subplot(2,3,6)
plt.imshow(cv2.imread(imgReq))
plt.title('Requete')
for i in range(1,nbFirst+1):
        #on commence à 1 pour éviter l'image requête qui est dans la base
        #cela doit être adapté si ce n'est pas le cas.
        n = ind[i]
        plt.subplot(2,3,i)
        filename = myImagePath + allNames[n]
        plt.imshow(cv2.imread(filename))
        plt.title('{}:{}'.format(i,allNames[n]))

plt.show()
```