UNIVERSITÉ NATIONALE DU VIETNAM À HANOÏ

Institut de la Francophonie pour l'Innovation

Indexation des contenus multimédia

Option : Systèmes Intelligents et Multimédia (SIM)

Promotion : 22

RAPPORT DU PROJET CIBR

Rédigé par :
OUBDA Raphaël Nicolas Wendyam

Enseignant :

Nicolas SIDERE

Année académique : 2018 - 2019

Table des matières

Table des figures

Liste des tableaux

1 Introduction

L'indexation automatique des documents multimédias a pour but de permettre, par le biais de techniques automatiques ou semi-automatiques, l'exploitation des collections de documents multimédias : textes, documents sonores, vidéos, télévision. Cette exploitation peut consister, par exemple, à rechercher un document donné dans une collection à partir d'une description ou d'un autre document, à naviguer dans une collection pour parcourir son contenu, à résumer la collection, à décrire automatiquement les documents pour les archiver, à utiliser ces descriptions pour produire de nouveaux documents ou de nouveaux services.L'exploitation desdites données nécessite au préalable leur traitement et plus précisément leur indexation afin de les structurer et donc de faciliter la recherche ultérieure d'informations.
Nous avons deux types d'indexation à savoir l'indexation textuelle et l'indexation basé sur le contenu visuel. L'indexation basé sur le contenu visuel fondé sur trois(3) approches :

- l'approche globale ;
- l'approche spatiale ;
- l'approche basé sur le global et l'approche spatial.

Dans le but de bien appréhender le cours d'indexation de contenus multimédia notre projet est basé sur la détection des similitudes d'images. Pour ce faire dans notre projet nous devons utiliser une indexation sur le contenu visuel et une approche global basé sur les descripteurs locaux des images(couleur, texture, forme).
Dans la suite de notre rapport nous allons détailler le dataset que nous avons utilisé, ensuite nous allons utilisé la couleur pour détecter les images similaires, dans la troisième partie nous allons utiliser la texture pour la détection, puis nous l'utilisation de la forme pour la détection nous terminerons en utilisant les trois descripteurs pour la détection des similitudes..

2 Description de la base d'images

Dans notre projet nous avons utilisé le dataset COIL(Columbia University Image Library). Ce dataset contient 100 différents objets avec 20 images pour chaque objet sous différentes prises de vues. Il est librement téléchargeable à l'adresse :
http ://www.cs.columbia.edu/CAVE/software/softlib/coil-100.php

FIGURE *1 – Aperçu du dataset images*

3 Utilisation de descripteurs globaux des images pour la recherche.

Dans cette partie nous allons utiliser les descripteurs des images pour la recherche des images. Ces descripteurs sont :

- la couleur : l'histogramme ;
- la texture : matrice de cooccurrence ;
- la forme : le moment de Hu.

Nous avons utilisé le langage python pour la réalisation de ce projet.

4 Système de recherche d'images en utilisant l'histogramme.

4.1 Étape pour la recherche.

Dans cette partie nous avons écris un programme en python qui est couleur.py .Pour ce faire nous avons écris différentes fonctions qui permettent de :

- calculer l'histogramme de chaque image ;

- stocker les histogrammes dans un fichier texte ;
- Calcul de la distance entre deux histogrammes ;
- retourner images similaires.
- fonction d'évaluation de l'algorithme.

4.1.1 Calcul de de l'histogramme

Dans cette partie nous avons utilisé la bibliothèque d'openCV pour extraire les histogrammes de chaque images. Nous avons réduit la taille des histogrammes afin de réduire le temps de calcul du programme. Les histogrammes sont récupérés sous forme de vecteur. La capture ci-dessous présente la fonction que nous avons écrit. La partie entouré en rouge représente la taille de l'histogramme.

```
#Calcul de l'histogramme et normalisation connaissant le chemin de l'image
def Histogramme(chemin):
    image = cv2.imread(chemin)
    histogramme = cv2.calcHist([image], [0, 1, 2], None, [16, 16, 16],[0, 256, 0, 256, 0, 256])
    histogramme = cv2.normalize(histogramme, histogramme).flatten()
    return histogramme
```

FIGURE *2 – Fonction de calcul d'histogramme.*

4.1.2 Stockage des histogrammes.

Dans cette partie après avoir calculer les histogrammes de toutes les images du dataset nous les stockons dans un fichier texte. Ainsi après le stockage des histogrammes nous n'avions plus besoins des images originaux. De ce fait il peuvent être supprimé. Notre fichier des histogramme se trouve dans le dossier train du projet et porte le nom **histogramme.txt**. Nous avons utilisé la fonction pickle de python pour le stockage cette fonction permet d'enregistrer ainsi les histogrammes de nos images sous forme d'objet. La capture ci-dessous présente la fonction utiliser pour permettre le stockage.

```
#Creation du fichier pour stockage des descripteurs pickle
def pickle_hist(fichier,histogramme):
    pkl=pickle.Pickler(fichier)
    pkl.dump(histogramme)

#Récupération du descripteur Unpickle
def unpickle_hist(fichier):
    Unpkl=pickle.Unpickler(fichier)
    fic=(Unpkl.load())
    return fic
```

FIGURE *3 – Fonction de stockage(pickle) des histogrammes.*

4.1.3 Calcul de la distance entre deux histogrammes d'une image.

Pour calculer la distance entre deux histogrammes, plusieurs distances existent notamment la distance euclidienne, la distance de Minkowski. Dans notre projet nous avons utilisé la distance

euclidienne dont la formule est la suivante :

$$\textbf{Distance(Hist1,Hist2)} = \sum_{i=0}^{3*M}((Hist1(i) - Hist2(i))^2$$

Notre fonction prend en entrée deux histogrammes et retourne la distance euclidienne entre les deux histogrammes. La capture ci-dessous présente notre fonction de calcul de distance.

```
#Fonction de calcul de distance entre deux histogrammes
def CalculDistance(hist1,hist2):
    distances = distance.euclidean(hist1,hist2)
    return distances
```

FIGURE 4 – *Fonction de calcul de distance entre histogramme.*

4.1.4 Fonction de retour des images similaires.

Cette fonction permet de retourner les images les plus proches, en utilisant la distance euclidienne. Ainsi quand nous renseignons une lien d'une image cette fonction permet de comparer les histogrammes de l'image d'entrée et les histogrammes du dataset stockés. Elle donne en sortie le nombre d'images similaires demander par l'utilisateur.

```
#Chercher les plus proches voisins
def RessemblaceImage(cheminImageTest,k):
    listeDistance ={}
    #Calcul de l'histogramme de l'image de test
    hist1 = Histogramme(cheminImageTest)
    hist1 = np.asanyarray(hist1)
    f = open((pathFichierTrain+"/histogramme"+".txt"),"rb")
    list_hist = unpickle_hist(f)
    for key, valeur in list_hist.items():
        valeur = np.asanyarray(valeur)
        CalculDistance(valeur,hist1)
        listeDistance[CalculDistance(valeur,hist1)] = key
    listeDistances = sorted(listeDistance.items(), key=lambda t: t[0])
    f.close
    return(listeDistances[:k])
```

FIGURE 5 – *Fonction de calcul des plus proches voisins.*

4.1.5 Évaluation du programme

Cette fonction de notre programme permet de calculer la précision, le rappel et la F-mesure. Selon les formules :

$$precision_i = \frac{Nombre\ \ de\ \ document\ \ correcte\ \ attribue\ \ \ \ la\ \ classe\ \ i}{Nombre\ \ de\ \ document\ \ attribue\ \ \ \ la\ \ classe\ \ i}$$

$$rappel_i = \frac{Nombre\ \ de\ \ document\ \ correcte\ \ attribue\ \ \ \ la\ \ classe\ \ i}{Nombre\ \ de\ \ document\ \ appartenant\ \ \ \ la\ \ classe\ \ i}$$

$$F - mesure = 2 * \frac{precision * rappel}{precision + rappel}$$

4.2 Test et analyse des résultats.

Pour lancer notre programme installer python. Puis lancer le fichier couleur.py, et suivez les instructions du programme. Le programmes prend en entrée le lien d'une image et le nombre d'image similaires à trouver et retourne les images similaires. Lorsque vous lancer le programme une interface.

4.2.1 Test et analyse avec obj1__10.png

Nous testons avec la taille de l'histogramme 8 et nous retournons 5 images les plus proches de l'obj1__10. La figure ci-dessous présente les résultats obtenus.

```
**********************************************************************
*                                                                    *
*         RECHERCHE D'IMAGE EN UTILISANT L'HISTOGRAMME               *

=> Entrer le chemin de l'image: ./obj1__10.png

=> Nombre d'image a cherché: 5

-------------LISTE des resssemblances---------------

Images                                Distance
obj1__10.png                          0.0
obj1__15.png                          0.02047639712691307
obj1__5.png                           0.024230679497122765
obj1__200.png                          0.03010161779820919
obj1__195.png                          0.030281873419880867

-------       EVALUATION     ---------
La précision est de =  1.0
Le rappel est de =  0.06944444444444445
La F-mesure est de =  0.12987012987012989
```

FIGURE *6 – 5 PPV de obj1__10.png.*

Analyse : Nous constatons que nous avons une précision de 1. De ce fait toutes les images retournées sont de même catégorie que l'image donnée en entrée. Nous effectuons un autre test avec la taille de l'histogramme 8 et nous recherchons les 30 images les plus proches. La figure ci-dessous présente les résultats obtenus.

```
------------LISTE des resssemblances--------------

Images                                  Distance
obj1__10.png                            0.0
obj1__15.png                            0.02047639712691307
obj1__5.png                             0.024230679497122765
obj1__200.png                           0.03010161779820919
obj1__195.png                           0.030281873419880867
obj1__0.png                             0.032905321568250656
obj1__20.png                            0.03450142592191696
obj1__355.png                           0.034758973866701126
obj1__350.png                           0.040847744792699814
obj1__25.png                            0.043810732662677765
obj1__190.png                           0.04491584002971649
obj1__205.png                           0.04751237854361534
obj1__345.png                           0.048811618238687515
obj1__30.png                            0.053235433995723724
obj84__25.png                           0.0543607659637928
obj1__340.png                           0.055080290883779526
obj1__210.png                           0.05781375244259834
obj1__335.png                           0.059335991740226746
obj1__300.png                           0.05945221334695816
obj1__175.png                           0.059882283210754395
obj1__35.png                            0.06045754253864288
obj1__330.png                           0.06257983297109604
obj1__305.png                           0.06287050247192383
obj1__295.png                           0.06434645503759384
obj1__325.png                           0.0647558644413948
obj1__310.png                           0.0648476779460907
obj1__315.png                           0.06516074389219284
obj1__320.png                           0.0656440481543541
obj84__30.png                           0.06570731103420258
obj1__120.png                           0.06580347567796707

-------       EVALUATION     ---------
La précision est de =  0.9333333333333333
Le rappel est de =  0.3888888888888889
La F-mesure est de =  0.5490196078431373
```

FIGURE 7 – *30 PPV de obj1__10.png.*

Analyse : Nous constatons que la précision est de 0.93 soit 93%. Deux images de catégories 84 ont été renvoyées, il s'agit des objets obj84__30.png et obj84__25.png. Ces faux positifs ont été renvoyés car leur histogramme est proche de celle de l'image d'entrée. Nous constatons que ces images ont la même couleur et la même forme que l'image d'entrée. De ce fait leur histogramme ont une similarité(Voir figure ci-dessous)

[obj1__10]   [obj84__30]  [obj84__25]

4.2.2 Test avec obj99__255.png

Nous testons ici avec une taille de l'histogramme 16 et nous voulons les 1es plus proches voisins. La figure ci-dessous présente le résultat obtenu.

```
****************************************************************
*                                                              *
*         RECHERCHE D'IMAGE EN UTILISANT L'HISTOGRAMME         *

=> Entrer le chemin de l'image: dataset/obj99__255.png

=> Nombre d'image a cherché: 10

-------------LISTE des resssemblances---------------

Images                                Distance
obj99__255.png                          0.0
obj99__250.png                          0.022899648174643517
obj99__285.png                          0.11532826721668243
obj99__150.png                          0.14074821770191193
obj99__160.png                          0.14753679931163788
obj99__200.png                          0.15188747644424438
obj99__175.png                          0.15326499938964844
obj84__295.png                          0.15327361226081848
obj99__165.png                          0.15994709730148315
obj1__100.png                           0.16589510440826416

-------       EVALUATION     ---------
La précision est de =  0.8
Le rappel est de =  0.1111111111111111
La F-mesure est de =  0.1951219512195122
```

FIGURE *8 – 10 PPV de obj99__255.png.*

Analyse : Nous constatons que obj1 et obj84 on été détecté comme plus proches de l'image obj99__255.png. Ces faux positifs ont été renvoyé car leur histogramme est proche de celle de l'image d'entrée. Nous constatons que ces images ont la même couleur et la même forme que l'image d'entrée. De ce fait leur histogramme ont similarité(Voir figure ci-dessous).
Nous avion une précision de 0.8 ce qui prouve que les résultats sont bonnes pour cette catégorie d'image. Le rappel est faible(0,11) car nous n'avions retourné que demandé 10 images sur 72 images de cette catégorie.

4.3 Bilan récapitulatifs des tests

Pour effectuer nos tests nos avons faits en plus des tests ci-dessus des tests avec différents paramètres c'est à dire la taille de l'histogramme, le nombre d'image à retourner le tableau ci dessous donne un bilan récapitulatif de nos tests.

TABLE *1 – Bilan récapitulatif des tests en utilisant l'histogramme*

Bilan récapitulatif					
Image	**T Histogramme**	**Nb images**	**Précision**	**Rappel**	**F-mesure**
obj1__10.png	8	30	0,93	0,39	0,54
	16	30	1	0,42	0,59
	16	72	0,65	0,65	0,65
obj84__25.png	8	30	0,87	0,36	0,51
	8	72	0,47	0,47	0,47
	16	30	0,87	0,36	0,51
	16	72	0,58	0,58	0,58

... suite de la page précédente...

Image	**T Histogramme**	**Nb images**	**Précision**	**Rappel**	**F-mesure**

... suite page suivante...

	8	30	1	0,42	0,59
obj35__255.png	8	72	1	1	1
	24	72	1	1	1
	24	72	1	1	1

Analyse

- Selon nos tests les résultats avec l'histogramme sont bonnes car la précision est toujours supérieur à 0.7 et pour certaines catégories nous avions eu un rappel de 1 de ce fait notre algorithme est efficace.
- Nous constatons que plus la taille de l'histogramme est grande plus la F-mesure est grande, mais le temps d'apprentissage augmente. Nous pouvons dire que la taille de l'histogramme a une influence sur notre système, sur le temps de calcul des histogramme ;
- Les catégories d'image qui ont une forme et une une couleur semblable ont été confondues (obj1__10 et obj84__25). Ainsi notre système renvoie des faux positifs si la taille de l'histogramme est petite et le nombre d'images est élevé cas de obj1__10 ;
- Pour les images qui ont une forme unique dans le dataset obj35__255 nous avons une bonne évaluation car on a une précision de 1 et un rappel de 1 donc toutes les images de la catégorie obj35 ont été renvoyées.

Conclusion : La recherche d'image en utilisant l'histogramme est efficace pour notre dataset car les images sont de petites tailles. De plus pour avoir une bonne évaluation avec cette technique la taille de l'histogramme doit tre grande ainsi nous considérons tous les pixels des images, mais cela influence le temps d'apprentissage qui est relativement long. Les images qui sont de catégorie différentes mais de même couleur sont confondu dont la recherche d'image n'est toujours pas efficace si les images du dataset ont toujours la même couleur.En effet, nous pouvons considérer la texture de l'image pour essayer de remédier à ce problème. De ce fait la recherche d'image en utilisant la texture constituera notre prochaine étude.
NB : Nous obtenons de meilleur résultat pour une taille de l'histogramme T=24

5 Système de recherche d'images en utilisant la texture : matrice de cooccurrence.

La matrice de cooccurrence est un descripteur de texture. Les matrices de cooccurrences consistent à identifier des motifs de pairs de pixels séparés par une distance d dans une direction i. Pour chaque couple (d, theta), construit sur une matrice (d, theta). Une matrice moyenne est alors calculée la rendant invariante à la rotation. Comme les matrices de cooccurrences contiennent

beaucoup d'informations et sont donc consommatrices en espace mémoire, elles ne sont pas directement exploitées. Les utilisateurs préfèrent donc extraire de ces matrices des attributs afin de réduire la quantité d'informations à manipuler, en conservant la pertinence de ces descripteurs. C'est pour cette raison que nous avons choisi d'utiliser les attributs suivants qui sont classiquement utilisés.

5.1 Explication du code.

Dans cette partie nous avons écris un programme en python qui est qui contenu dans le fichier cooccurrence.py du projet. Pour ce faire nous avons écris différentes fonctions dans ce programme qui permettent de rechercher les images en fonctiion de la matrice de cooccurrence. Ces fonctions sont :

- transformer l'image en niveau de gris
- Normalisation ;
- Calcul des paramètres de matrice de cooccurrence ;
- stockage des paramètres dans un fichier texte ;
- Calcul de la distance entre les paramètres de deux images.

NB : A la fin de ces étapes nous n'avions plus besoins du dataset car nous avions stocké les vecteurs dans un fichier texte qui est cooccurrence contenu dans le dossier train de notre projet. De ce fait le dataset n'est plus nécessaire.

5.1.1 Transformation des images en niveau de gris.

Nous calculerons les textures uniquement sur l'image en niveau de gris pour simplifier la tâche. Pour convertir les images en niveaux de gris il suffit d'utiliser (pour chaque pixel) la fonction suivante :

$$\textbf{gris} = (\textbf{rouge} + \textbf{vert} + \textbf{bleu})/3$$

Pour ce faire nous avons utilisé une fonction de opencv. La capture ci-dessous présente cette fonction :

```
#Transformation d'une image en niveau de gris
def Gris(chemin):
    image = cv2.imread(chemin)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    gray = normalisationImage(gray)
    return gray
```

FIGURE *9 – Fonction de transformation en niveau de gris.*

5.1.2 Normalisation

Nous réduirons la quantification de l'image pour la faire passer de 256 à T niveaux de gris (T=8, 16 ou 24). La fonction de normalisation est la suivante :

```python
#Normalisation de l'image
def normalisationImage(image):
    normImage = image/8
    normImage = normImage.astype('uint32')
    return normImage
```

FIGURE *10 – Fonction de normalisation.*

5.1.3 Calcul de la matrice de cooccurrence

Dans cette partie nous avons écris une fonction qui prend en entrée une image en niveau de gris et retourne la matrice de cooccurrence. La capture ci-dessous présente la fonction que nous avions écrit en python.

```python
#Matrice de co-occurence
def MatCooccurence(image_gris):
    matCo = greycomatrix(image_gris, [5], [0,np.pi/2,np.pi/4,(np.pi*3)/4], 256,
                         symmetric=True, normed=True)
    return matCo
```

FIGURE *11 – Fonction pour le calcul de matrice de cooccurrence.*

5.1.4 Calcul des paramètres de la matrice de cooccurrence

Dans cette partie nous avions écrit une fonction qui permet de calculer les paramètres de matrices de cooccurrence. Les formules de cette matrice sont représentés sous la figure suivante :

$$ENERGIE = \sum_j \sum_i (p(i,j))^2$$

$$INERTIE = \sum_j \sum_i (i-j)^2 \, p(i,j)$$

$$ENTROPIE = -\sum_j \sum_i p(i,j) \log(p(i,j))$$

$$MOMENT\ DIFFERENTIEL\ INVERSE = \sum_i \sum_j \frac{1}{1+(i-j)^2} p(i,j)$$

FIGURE *12 – Formule des paramètres de matrice de cooccurence.*

Nous avions écrit la fonction en python elle prend en entrée une matrice de cooccurrence et retourne les paramètres suivantes :

- l'énergie ;

- la disimilarité ;
- le contraste ;
- l'homogénéité ;
- la corrélation.

Ainsi la figure ci-dessous présente notre fonction python.

```python
#Calcul des parametre de la co occurence
def ParamCooccurence(HistomatCoo):
    #Calcul de l'energie
    energie = greycoprops(HistomatCoo,'energy')
    contraste = greycoprops(HistomatCoo,'contrast')
    dissimilarite = greycoprops(HistomatCoo,'dissimilarity')
    homogeneite = greycoprops(HistomatCoo,'homogeneity')
    correlation = greycoprops(HistomatCoo,'correlation')
    return energie, contraste, dissimilarite,homogeneite,correlation
```

FIGURE *13 – Fonction qui retourne les paramètres de matrice de cooccurence.*

Nous calculons la matrice de cooccurence de toutes les images du dataset et nous les stockons dans un fichier texte. Ainsi après le stockage nous n'avions plus besoin des images car les descripteurs sont déja enregistrés dans un fichier à l'aide de la fonction pickle de python.

5.1.5 Calcul de la distance entre les paramètres.

Pour renvoyer les K plus proches voisins des images requête nous avions écrit une fonction en python qui permet de calculer la distance entre les paramètres de deux matrices de cooccurence. La formule de calcul de la distance est la suivante :

$$\text{Distance(im1,im2)} = \sqrt{(\sum_{i=1}^{5}(param_i(im1) - param_i(im2))^2))/5}$$

La capture ci-dessous présente la fonction de calcul de la distance.

```python
#Calcul de la distance
def CalculDistance(image1,image2):
    d = (np.linalg.norm((image1-image2))/5)
    return d
```

FIGURE *14 – Fonction de calcul de la distance entre les paramètres de la matrice de cooccurence.*

Une autre fonction calcule les distances entre les images du dataset et l'image en entrée et renvoie les K plus proches voisin de l'image requête.

```python
#Recherche des ressemblance
def Ressemblance(chemin,k):
    listeDistance ={}
    param = np.zeros(5)
    image_gris = Gris(chemin)
    MatCoo = MatCooccurence(image_gris)
    energie,contraste,dissimilarite,homogeneite,correlation = ParamCooccurence(MatCoo)
    energie = energie[0][0]
    param[0] = energie
    contraste = contraste[0][0]
    param[1] = contraste
    dissimilarite = dissimilarite[0][0]
    param[2] = dissimilarite
    homogeneite = homogeneite[0][0]
    param[3] = homogeneite
    correlation = correlation[0][0]
    param[4] = correlation
    f = open((pathFichierTrain+"/cooccurence"+".txt"),"rb")
    list_hist = couleur.unpickle_hist(f)
    for key, valeur in list_hist.items():
        valeur = np.asanyarray(valeur)
        listeDistance[CalculDistance(valeur,param)] = key
    listeDistances = sorted(listeDistance.items(), key=lambda t: t[0])
    f.close
    return(listeDistances[:k])
```

FIGURE *15 – Fonction qui retourne les plus proche voisin en utilisant la matrice de cooccurrence.*

5.2 Test et analyse des résultats avec la matrice cooccurrence

Pour lancer le programme aller dans le dossier du projet puis taper **python cooccurrence.py**. Une interface apparaît et demande de renseigner le lien de l'image requête et le nombre d'image à retournée et le programme vous renvoie les K plus proches voisins et l'évaluation correspondante.

5.2.1 Test et évaluation avec obj11__295.png

Pour l'évaluation nous avons normalisé avec une valeur de normalisation T=8 et nous retournons les dix(10) images les plus proches de l'image d'entrée. La capture ci-dessous présente le résultat :

```
**********************************************************************

             DISTANCE EN UTILLISANT LA MATRICE DE COOCCURRENCE

Entrer le chemin de l'image: dataset/obj11__295.png

Entrer le nombre de ressemblances à trouver: 10
----------------------------------------------------------


------------LISTE des resssemblances en utilisant la cooccurence-

Images                                Distance
obj11__295.png                          0.0
obj11__290.png                          0.009579659323521523
obj11__255.png                          0.013432961944022853
obj4__55.png                          0.013470011317544698
obj11__105.png                          0.014179322252384519
obj11__285.png                          0.014761449604488395
obj42__285.png                          0.01635906976107402
obj11__280.png                          0.016542221552668393
obj11__85.png                         0.017329680643548118
obj11__260.png                          0.018651117265385254

-------         EVALUATION     ---------
La précision est de =  0.8
Le rappel est de =  0.1111111111111111
La F-mesure est de =  0.1951219512195122
********     FIN      ***********
```

FIGURE *16 – 10 PPV de obj11__295.*

Analyse : le système retourne les 10 plus proches voisins avec une précision de 0.8 et un rappel de 0.19. Une image de obj42__285.png, cette image est renvoyée car elle a une texture semblable

à l'objet obj11__295. Le rappel est faible car dans le dataset nous avions 72 images pour chaque catégorie et toutes les images de cette catégorie n'ont pas été trouvé de ce fait le rappel est faible.

[obj11__2]   [obj47__285]

5.2.2 Test et évaluation avec obj69__35.png

Pour l'évaluation nous avons normalisé avec une valeur de normalisation T=8 et nous retournons les vingt(10) les plus proches de obj69__35.png. La capture ci-dessous présente le résultat :

```
-----------LISTE des resssemblances en utilisant la cooccurence-

obj69__35.png                          0.0
obj69__30.png                          0.001773502935439508
obj69__20.png                          0.005913070512208042
obj15__355.png                          0.006562894486461887
obj91__170.png                          0.006935088681253912
obj15__140.png                          0.007375319450923942
obj69__15.png                          0.008454131907508444
obj15__0.png                          0.008476997153050753
obj91__160.png                          0.009549749358533022
obj23__325.png                          0.010124884104306477
obj91__155.png                          0.010651054136687679
obj91__180.png                          0.011021592898987898
obj67__20.png                          0.011572943672470899
obj91__200.png                          0.011956245028110369
obj91__355.png                          0.01226124125422384
obj91__175.png                          0.013218672646767394
obj69__25.png                          0.013749843601928149
obj76__210.png                          0.014093635485188594
obj15__170.png                          0.014271786700525837
obj21__225.png                          0.014349196016848855

-------         EVALUATION     ---------
La précision est de =  0.25
Le rappel est de =  0.06944444444444445
La F-mesure est de =  0.10869565217391305
********     FIN      ***********
```

FIGURE *17 – 10 PPV de obj69__35.png.*

Analyse : Nous constatons que plusieurs images de obj91, obj15, obj11 ont été confondues avec la catégorie obj69 cela est du au faite que les images ont une texture similaire de ce fait leurs matrice de cooccurence sont proches. Ainsi, plus K est élevé plus le système renvoie des faux positifs. Les images ci dessous sont de catégorie différentes, mais elles ont été confondues car elles ont une même texture et le même objet sur l'image qui est une voiture.

[obj15]   [obj91]  [obj69]

5.3 Bilan récapitulatifs des tests avec la matrice de cooccurrence

Pour effectuer nos tests nos avons faits en plus des tests ci-dessus des tests avec différents paramètres c'est à dire la normalisation, le nombre d'image à retourner le tableau ci dessous donne un bilan récapitulatif de nos tests.

TABLE *2 – Bilan récapitulatif des tests avec la matrice de cooccurrence*

Bilan récapitulatif					
Image	**Normalisation**	**Nb images**	**Précision**	**Rappel**	**F-mesure**
obj75__255.png	8	20	0,7	0,19	0,3
	8	70	0,47	0,37	0,38
	16	20	0,9	0,25	0,39
	16	70	0,47	0,46	0,46
obj47__255.png	8	20	1	0,28	0,43
	8	70	0,73	0,71	0,72
	16	20	1	0,28	0,44
	16	70	1	0,07	0,98

NB : la précision,le rappel et la F-mesure ont été calculés sur une échelle de 1.

Analyse : Nous constatons que notre algorithme est efficace car la précision et le rappel que nous obtenons sont assez bonnes. Mais des images de certains catégories d'image sont ce pendant confondu, c'est le cas des objets obj69, obj11, obj91. Ces faux positifs sont du au fait qu'ils ont une texture similaire ainsi plus K plus proches voisins est élevé plus le système renvoie de mauvaises images. Nus constatons que la valeur de la normalisation a une influence sur l'algorithme nous constatons qu'avec une normalisation de T=8 les résultats sont meilleurs par rapport à une normalisation de T = 16 car 16 nous avons moins de pixels donc il y'a perte d'information. Nous constatons que plus T de la normalisation est grande plus le temps d'apprentissage est relativement faible car cela réduit le nombre de pixel à traiter. **NB : Nous obtenons de meilleur résultat pour T = 8.**

6 Système de recherche d'images en utilisant la forme : moment de HU.

6.1 Explication du code de recherche en utilisant les moments de HU

La troisième caractéristique que nous allons utiliser pour comparer 2 images est un descripteur de forme : les moments de Hu. Pour chaque image, ces moments vont résumer la forme de l'objet. Nous avions écrit un programme en python qui permet de rechercher ces images. Le programme se trouve dans le fichier **moment.py** du projet. Pour ce faire nous avons suivi les étapes suivantes :

- Conversion en niveau de gris ;
- Réduction des niveaux de gris de l'image ;

- Calcul des moments et des caractéristiques ;
- stockage des moments de HU ;
- calcul de la distance entre deux moments.

6.1.1 Conversion en niveau de gris

Les moments de Hu se calcule sur des images en niveau de gris donc cette phase est primordiale. De ce fait nous avions écrit une fonction en python. La capture ci-dessous présente cette fonction :

```python
#Transformation d'une image en niveau de gris
def Gris(chemin):
    gray = cv2.imread(chemin,cv2.IMREAD_GRAYSCALE)
    _,im = cv2.threshold(gray, 128, 255, cv2.THRESH_BINARY)
    return im
```

FIGURE *18 – Fonction de conversion en niveau de gris.*

6.1.2 Réduction en niveau de gris

Pour réduire le temps de calcul, nous avions réduit le niveau de gris des images du dataset. La capture ci-dessous présente notre fonction en python :

```python
#Normalisation de l'image
def normalisationImage(image):
    normImage = image//8
    normImage = normImage.astype('uint32')
    return normImage
```

FIGURE *19 – Fonction de réduction en niveau de gris.*

6.1.3 Calcul des moments de HU

Ils sont calculés à partir des images en niveaux de gris. Ainsi pour calculer les moments de Hu d'une image nous la convertissons d'abord en niveau de gris et ensuite nous une réduction de niveau de gris de l'image. Toutes ces étapes ont déjà été expliquées ci-dessous. Grâce à la fonction HuMoments() de openCV nous calculons les moments Hu. La capture ci-dessous présente la fonction :

```python
#Calcul des moment de HU
def momentHu(image):
    moments = cv2.moments(image)
    huMoments = cv2.HuMoments(moments)
    return huMoments
```

FIGURE *20 – Fonction de calcul du moment de HU.*

6.1.4 Stockage des moments de HU

Après avoir calculer les moments de HU nous les stockons dans un fichier texte. Ainsi les images du dataset ne sont plus nécessaires. Pour cela nous avions écrit une fonction en python à l'aide

de la fonction pickle dont présente la figure ci-dessous :

```
#Creation du fichier pour stockage des descripteurs pickle
def pickle_hist(fichier,histogramme):
    pkl=pickle.Pickler(fichier)
    pkl.dump(histogramme)

#Récupération du descripteur Unpickle
def unpickle_hist(fichier):
    Unpkl=pickle.Unpickler(fichier)
    fic=(Unpkl.load())
    return fic
```

FIGURE *21 – Fonction de stockage.*

6.1.5 Calcul de la distance entre les moments de HU

Pour renvoyer les K images les plus proches nous avions écrit une fonction en python qui calcul la distance entre le moment de Hu de l'image d'entrée et les moments de Hu du dataset. La capture ci-dessous présente cette fonction :

```
#Fonction de calcul de distance euclidienne entre deux moment de Hu
def CalculDistance(moment1,moment2):
    distances = distance.euclidean(moment1,moment2)
    return distances
```

FIGURE *22 – Fonction de calcul des distances.*

6.1.6 Recherche des ressemblance en utilisant les moments de HU

Dans cette partie nous avions écrit une fonction qui prend en entrée le lien d'une image requête et le nombre d'image K à rechercher puis il retourne les images similaires. La fonction calcul le moment de Hu de l'image d'entrée et le compare au moment de Hu des images du dataset stocké stocké déjà dans notre fichier **moment.txt**. Cette comparaison est faite en calculant la distance entre le moment de Hu de l'image requête et les moments de Hu des images du dataset stockés dans le fichier. Le système retourne les K images les plus proches. La capture ci-dessous présente notre fonction :

```python
#Chercher les plus proches voisins
def RessemblaceImage(cheminImageTest,k):
    listeDistance ={}
    #Calcul des moments de hu de l'image de test
    gris = Gris(cheminImageTest)
    moment_test = momentHu(gris)
    f = open((pathFichierTrain+"/moment"+".txt"),"rb")
    list_hist = couleur.unpickle_hist(f)
    for key, valeur in list_hist.items():
        d = CalculDistance(valeur,moment_test)
        listeDistance[d] = key
    listeDistances = sorted(listeDistance.items(), key=lambda t:t[0])
    f.close

    return(listeDistances[:k])
```

FIGURE *23 – Fonction de réduction en niveau de gris.*

6.2 Test et évaluation.

Pour évaluer notre algorithme de recherche d'images basé sur le moment de Hu nous allons effectuer des tests. Pour lancer le programme aller dans le dossier du projet puis taper **python moment.py**.Une interface apparaît et demande de renseigner le lien de l'image requête et le nombre d'image à retournée et le programme vous renvoie les K plus proches voisins et l'évaluation correspondante.

6.2.1 Test avec obj2__250

Nous testons avec obj2__250 en prenant K=10. La figure ci-dessous présente le résultat obtenu :

```
**********************************************************************

              DISTANCE EN UTILLISANT LA FORME (MOMENT DE HU)

Entrer le chemin de l'image: dataset/obj2__250.png

Nombre d'image a cherché: 10

-------------LISTE des resssemblances---------------

Images                                  Distance
obj2__250.png                            0.0
obj2__15.png                            1.4634913933825833e-08
obj2__20.png                            4.978504144197096e-08
obj2__205.png                            3.10740853850418e-07
obj2__200.png                            1.1589867849092775e-06
obj2__245.png                            1.3581587254108265e-06
obj2__30.png                            1.416659984267592e-06
obj65__160.png                            1.748645484561267e-06
obj2__25.png                            1.7601122927788388e-06
obj2__240.png                            1.8016959308252645e-06

-------      EVALUATION      ---------
La précision est de =  0.9
Le rappel est de =  0.125
La F-mesure est de =  0.21951219512195122
```

FIGURE *24 – 10 PPV de obj2__250*

Analyse : Nous avons une précision de O.9 sur une échelle de 1 ce qui un bon résultat car sur les dix image recherchées une seule n'est pas de la même catégorie que l'image d'entrée. Il s'agit notamment de obj65__165. Le rappel n'est pas bon car plusieurs images similaires sont toujours dans le dataset.

- K=30, la figure ci-dessous présente le résultat obtenu

```
obj2__10.png                          1.9235926684220526e-06
obj2__40.png                          2.254866265770346e-06
obj48__125.png                          2.32578429857957e-06
obj48__100.png                          2.684277729482152e-06
obj2__35.png                          3.0206683841859773e-06
obj65__200.png                          3.0364583173708005e-06
obj2__210.png                         3.615528147699012e-06
obj2__195.png                         4.092067657328296e-06
obj2__255.png                         4.1341689467985e-06
obj65__165.png                          4.389079640760727e-06
obj2__5.png                           5.080589891416375e-06
obj2__45.png                          5.593358813045973e-06
obj2__50.png                          5.914925623274743e-06
obj48__105.png                          6.116647558349608e-06
obj2__190.png                         6.407811099925231e-06
obj2__0.png                           6.696994050054462e-06
obj2__235.png                         6.7016954083311806e-06
obj2__215.png                         6.895024647745262e-06
obj2__260.png                         7.0318525049152835e-06
obj2__220.png                         7.209236287175535e-06

-------      EVALUATION      ---------
La précision est de =  0.8
Le rappel est de =  0.3333333333333333
La F-mesure est de =  0.47058823529411764
```

FIGURE *25 – 30 PPV de obj2__250*

- K=72

```
obj2__60.png                          1.3654359309517998e-05
obj2__350.png                          1.3962541368669655e-05
obj48__265.png                          1.4105483831344214e-05
obj2__345.png                         1.4607743441744941e-05
obj2__180.png                         1.4988786197335249e-05
obj48__90.png                         1.548683662199059e-05
obj48__280.png                          1.5757453709743073e-05
obj48__115.png                          1.6096328652263918e-05
obj2__340.png                         1.6288642673449347e-05
obj2__275.png                         1.6463132127788516e-05
obj57__115.png                          1.655454368501351e-05
obj2__335.png                         1.667499687668185e-05
obj2__65.png                          1.6807898182917476e-05
obj48__270.png                          1.723880017758909e-05
obj2__325.png                         1.741789560884753e-05
obj2__330.png                         1.7428517276104217e-05
obj2__70.png                          1.789774008352426e-05
obj2__175.png                         1.8240948432928604e-05
obj2__320.png                         1.8338321252721324e-05
obj2__75.png                          1.8496860525770274e-05
obj2__280.png                          1.9061153867195802e-05
obj57__125.png                          1.9339159304506275e-05
obj2__80.png                          2.128257376398323e-05

-------      EVALUATION      ---------
La précision est de =  0.6527777777777778
Le rappel est de =  0.6527777777777778
La F-mesure est de =  0.6527777777777778
```

FIGURE *26 – 72 PPV de obj2__250*

Analyse : Nous constatons que l'algorithme donne de bon résultat pour cette image. La précision est assez bonne mais lorsque nous augmentons le K des image de classe différentes sont détectées il s'agit de objet 57 et 48. Ces images sont détectées car elles ont une forme similaire à l'image d'entrée(voir figure ci-dessous).

[obj2__250]   [obj48]

Conclusion : Les images ci-dessous ont été confondues car elles ont une forme qui présente des similitudes, or notre recherche est basé sur la forme de ce fait les images qui ont une forme similaire mais de catégorie différentes sont confondues lorsque le K est élevé.

6.2.2 Test avec obj27__10

Nous testons sur l'objet 27 avec k=20

```
**************************************************************

            DISTANCE EN UTILLISANT LA FORME (MOMENT DE HU)

Entrer le chemin de l'image: dataset/obj27__10.png

Nombre d'image a cherché: 20

-------------LISTE des resssemblances---------------

Images                                  Distance
obj27__10.png                            0.0
obj29__290.png                           2.412771821429327e-06
obj32__115.png                           5.1378474426095115e-06
obj29__115.png                           9.733392203658087e-06
obj62__45.png                           1.1000437965847758e-05
obj47__180.png                           1.1107977956517338e-05
obj47__315.png                           1.1218782864338871e-05
obj27__85.png                           1.1473905799431087e-05
obj90__150.png                           1.1799815862490575e-05
obj52__25.png                           1.2335366384473262e-05
obj47__185.png                           1.2683307523702216e-05
obj47__300.png                           1.431757189427318e-05
obj23__330.png                           1.457775425294605e-05
obj81__305.png                           1.4807124395965643e-05
obj78__350.png                           1.625124750049153e-05
obj98__355.png                           1.6406857329310533e-05
obj90__160.png                           1.6523908512245734e-05
obj29__230.png                           1.7521499929616405e-05
obj93__305.png                           1.9522009400378347e-05
obj90__340.png                           1.9707729202874478e-05

-------      EVALUATION      ---------
La précision est de =  0.1
Le rappel est de =  0.027777777777777776
La F-mesure est de =  0.043478260869565216
```

FIGURE *27 – 20 PPV de obj27__10*

- K=70

Analyse : Nous constatons que les résultats son très mauvaises pour certain catégorie d'image notamment obj27, obj1, obj47.. Cela est du au faite que dans le dataset beaucoup d'image ont des formes similaires de ce fait notre système confond les catégorie car le programme a été écrit en utilisant le moment de Hu qui est un descripteur basé sur la forme.

6.2.3 Bilan des tests avec le moment de Hu

Pour effectuer nos tests nos avons faits en plus des tests ci-dessus des tests avec différents paramètres c'est à dire la normalisation, le nombre d'image à retourner le tableau ci dessous donne un bilan récapitulatif de nos tests.

TABLE *3 – Bilan récapitulatif des tests avec le moment de Hu*

Bilan récapitulatif				
Image	**Nb images**	**Précision**	**Rappel**	**F-mesure**
obj75__255.png	10	0,1	0,014	0,02
	20	0,05	0,014	0,02
	70	0,01	0,014	0,02

... suite page suivante...

... suite de la page précédente...

Image	**Nb images**	**Précision**	**Rappel**	**F-mesure**
obj47__255.png	10	0,5	0,07	0,12
	20	0,4	0,11	0,17
	70	0,27	0,26	0,26
obj100__25.png	10	0,2	0,028	0,049
	20	0,15	0,04	0,065
	70	0,1	0,1	0,1
	100	0,09	0,125	0,1

Analyse : Nous constatons que les résutats avec le moment de Hu ne sont pas assez bonnes car les images de notre dataset ont toujours une similitude en terme de forme avec d'autres images de ce fait le système confond les différents catégories des images d'ou les résultats moyens selon le tableau ci dessus. Les images ci dessous présentent quelque images de différentes catégories ayant la même forme.

[obj62]   [obj93]  [obj49]

Les images ci dessous ont la même forme mais de catégorie différentes. Ainsi notre système de recherche qui est basé sur la forme confond ces catégories. Les obj62, obj93, obj49 ont la même forme donc ils sont confondus.

[obj78]   [obj42]  [obj38]

[obj45]   [obj43]  [obj16]

Ainsi nous constatons que les catégories obj45, obj43, obj16 qui sont des tasses ont la même forme. De même que les catégories obj38, obj42 et obj78. Ainsi notre système confond les images qui ont les mêmes formes car les moments de Hu sont des descripteurs de formes.

7 Système de recherche d'images en utilisant les trois descripteurs.

7.1 Explication du code.

Dans cette partie nous avions écrit notre programme en python qui se trouve dans le fichier **app.py**. Nous recherchons les images en utilisant les trois descripteurs c'est-à-dire l'histogramme, la matrice de cooccurence et le moment de Hu. Pour ce faire chaque valeur du descripteur a un poids :

- w1 : pour l'histogramme ;
- w2 : pour la matrice de cooccurrence ;
- w3 : pour le moment de Hu

Ainsi nous calculons les distances entre les descripteurs de l'image d'entrée et les descripteurs du dataset stocké dans le dossier train.

7.2 Test, évaluation et analyse des résultats en utilisant les trois descripteurs.

Pour lancer le programme taper **python app.py**. L'interface du programme apparaît et vous donne la possibilité d'utiliser le système de recherche en utilisant le descripteur de votre choix. Pour utiliser les trois descripteurs nous choisissons l'option 4.

```
**********************************************************************
* ====================> Auteur: WORN 777 <==========================*
*    1-Similarité en utilisant la couleur.(Histogramme)              *
*    2-Similarité en utilisant la texture(matrice de cooccurrence)   *
*    3-Similarité en utilisant la forme(Moment de Hu)                *
*    4-Fonction gloabale de similarité entre 2 images                *
*                                                                    *
**********************************************************************

    Votre choix: 4
```

FIGURE *28 – Interface du système de recherche.*

Puis nous renseignons le nombre d'image à recherché,le lien de l'image et les poids w1, w2, w3

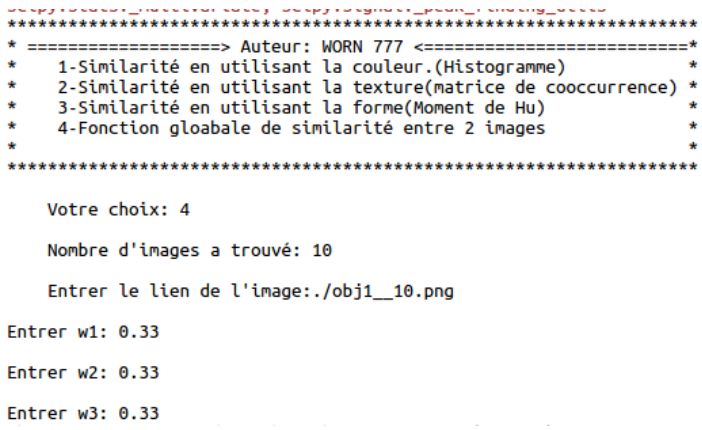

FIGURE *29 – Interface du système de recherche 1.*

Nous effectuons différents tests afin de trouver les meilleurs poids des w pour une bonne recherche c'est-à-dire pour avoir de meilleur résultat. Pour ce faire nous faisons varier les poids w. En effet, le tableau ci-dessous présente les résultats que nous avions obtenus :

TABLE *4 – Bilan récapitulatif des tests avec les trois descripteurs*

Bilan récapitulatif							
Image	**Nb images**	w1	w2	w3	**Précision**	**Rappel**	**F-mesure**
obj1__15.png	10	0,33	0,33	0,33	0,9	0,125	0,125
	10	0,6	O,3	0,1	1	0,14	0,14
	20	0,33	0,33	0,33	0,65	0,18	0,28
	20	0,6	0,3	0,1	0,85	0,23	0,36
	10	0,3	0,1	0,6	0,9	0,25	0,39
obj17__10.png	20	0,6	0,3	0,1	1	0,27	0,43
	20	0,3	0,1	0,6	1	0,27	0,43
	72	0,3	0,1	0,6	0,9	0,9	0,9
obj8__10.png	72	0,33	0,33	0,33	0.16	0,18	0,18
	72	0,6	0,3	0,1	0.26	0,26	0,26
	72	0,3	0,6	0,1	0,11	0,11	0,11
	72	0,3	0,1	0,6	0,375	0,375	0,375
	72	0,33	0,33	0,33	0.33	0,33	0,33

... suite page suivante...

... suite de la page précédente...

Image	**Nb images**	w1	w2	w3	**Précision**	**Rappel**	**F-mesure**
obj98_ _10.png	100	0,33	0,33	0,33	0.27	0,375	0,375
	72	0,6	0,3	0,1	0,375	0,375	0,375
	100	0,6	0,3	0,1	0,375	0,375	0,375
	100	0,3	0,6	0,1	0,27	0,27	0,27
	72	0,3	0,1	0,6	0,4	0,4	0,4
	100	0,3	0,1	0,6	0,31	0,43	0,36

Conclusion : Nous constatons que pour une variation des poids w la précision et le rappel varie. Pour ce faire nous avons une bonne évaluation pour les valeur :

- w1(couleur) = 0.3 ;
- w2 (texture) = 0.1 ;
- w3(forme) = 0.6.

De ce fait pour avoir une bonne évaluation il faut que le poids du moment de Hu soit élevé. Nous une bonne évaluation il faut plus miser sur la forme de l'image et la couleur, la texture n'impacte pas l'évaluation.

8 Conclusion

Dans ce projet de recherche d'image en utilisant les descripteurs locaux, nous avions écrit tous nos programme en python. Les descripteurs des images sont stockés dans un dossier train de notre projet. Nous avions faits l'étude en utilisant trois descripteurs :

- descripteur de couleur(l'histogramme) nous obtenons une meilleur évaluation de notre système avec une taille de l'histogramme à $\mathbf{T = 24}$;
- descripteur de texture(matrice de cooccurrence) nous obtenons une meilleur évaluation pour une normalisation avec la valeur **8** ;
- descripteur de forme(moment de Hu).

Nous avions combiné les trois descripteurs chacun ayant un poids avec **w1+w2+w3=1**. Ainsi nous avions évaluer notre système en faisant varier les différents poids de w. Nous obtenons de meilleur résultat avec les valeur **w1=0.3, w2=0.1 et w3=0.6**. Ainsi nous pouvons en conclure que pour obtenir de bon résultat avec les trois descripteurs il faut w2<w1<w3 et w1+w2+w3=1. Notre évaluation s'est basé sur le calcul de la précision, du rappel et de la F-mesure. Pour améliorer les résultats, nous pourrions utiliser le détecteur de SIFT ou Haris qui permettent la détection des points d'intérêts d'une image ou CNN pour la détection des caractéristiques des images ainsi que des algorithmes d'optimisation telle que K-means.

9 Références

https ://www.kaggle.com/andyxie/k-means-clustering-implementation-in-python
https ://www.learnopencv.com/shape-matching-using-hu-moments-c-python/
https ://www.learnopencv.com/shape-matching-using-hu-moments-c-python/