



INDEXATION MULTIMODALE DES CONTENUS

M-2, P-20- IFI 2016

Enseignant: Mickaël Coustaty

Rapport de Projet

Recherche dans de Grandes Bases d'Images

Base d'images Utilisée: COIL-100

Étudiants: Ginel Dorleon

Gervais Fotsing Sikadie S.

Hanoï, Mai 2017

Table des indices

INDEXATION MULTIMODALE DES CONTENUS M20, P-20- IFI 2016

Enseignant: Mickaël Coustaty

Rapport de Projet

INTRODUCTION.....	3
Première Partie -.....	4
I - Approche de Conception.....	5
II - Descripteurs Globaux sur les images.....	6
1 - Descripteurs Globaux sur les images – La couleur.....	6
A - Histogramme.....	6
B- Moments de HU.....	7
2 - Texture.....	7
3 - Forme.....	7
III - IMPLÉMENTATION.....	8
1. Description Globale Du Contenu Des Images.....	8
2. Histogrammes couleurs quantifiés.....	8
3. Intersection de deux histogrammes.....	9
4. Les moments de Hu.....	10
a - Conversion de l'image en degrés de gris.....	10
b - Réduction de l'image en niveaux de gris.....	10
c - Calculs des moments de Hu.....	10
d - Calcul de la distance entre deux images différentes basées sur les moments de Hu.....	10
5. Système de Recherche d'images.....	11
6. Fonction Globale de Similarité entre deux images.....	11
7. Clustering.....	11
III - EXPÉRIENCE ET RÉSULTATS.....	11
IV- CONCLUSION DE LA PREMIÈRE PARTIE.....	20
Deuxième Partie -.....	21
Utilisations des Descripteurs Locaux et Approche Bag-of-Words.....	21
I - GÉNÉRALITÉ.....	22
1 - Des régions d'intérêt.....	22
II - IMPLÉMENTATION.....	22
1. PHASE D' APPRENTISSAGE.....	22
a - Détection des points d'intérêts.....	22
b - Calcul d'un descripteur local pour chacun des keypoints.....	23
2 - Définition du dictionnaire de mots visuels, BOW.....	23
a - Implémentation du BOW.....	23
III- EXPÉRIENCE ET RÉSULTATS.....	25
IV - CONCLUSION BAGS-OF-VISUAL-WORDS.....	26
ÉVALUATION DU SYSTÈME	27
1 - Rappel.....	27
2- Précision.....	27
I - RÉSULTATS DE L'ÉVALUATION.....	28
II – DISCUSSION, CONCLUSION GÉNÉRALE ET PERSPECTIVE.....	29

I - INTRODUCTION

La recherche d'images par le contenu est un domaine de recherche très actif depuis plusieurs années. De nos jours, avec l'explosion de l'imagerie numérique, on se retrouve très souvent face à de grandes masses de données, ceci étant, la recherche d'une image particulière devient un problème délicat à résoudre.

Aujourd'hui, des études ont montré que, l'une des solutions la plus efficace est d'annoter manuellement ces images avec des mots-clés. Ce processus d'annotation représente un travail long et difficile accompagné de plusieurs approches.

Dans le cadre de ce projet, nous allons implémenter un système de recherche d'images par le contenu en se focalisant sur deux approches différentes: les descripteurs globaux et les descripteurs locaux.

L'idée alors est de calculer les caractéristiques pour chaque image en entrée et de rechercher les images caractéristiques les plus semblables.

Ainsi dans ce rapport, nous allons présenter le travail détaillé réalisé et les différentes méthodes utilisées, ensuite les résultats obtenus. En annexe, nous donnerons les programmes implémentés.

Alors, ce travail qui est divisé en deux grandes parties. Une première partie qui concerne l'utilisation des descripteurs globaux des images et une deuxième partie qui elle même concerne les descripteurs locaux et une approche du modèle bag-of-words.

- Première Partie -

Utilisation des descripteurs globaux sur des images

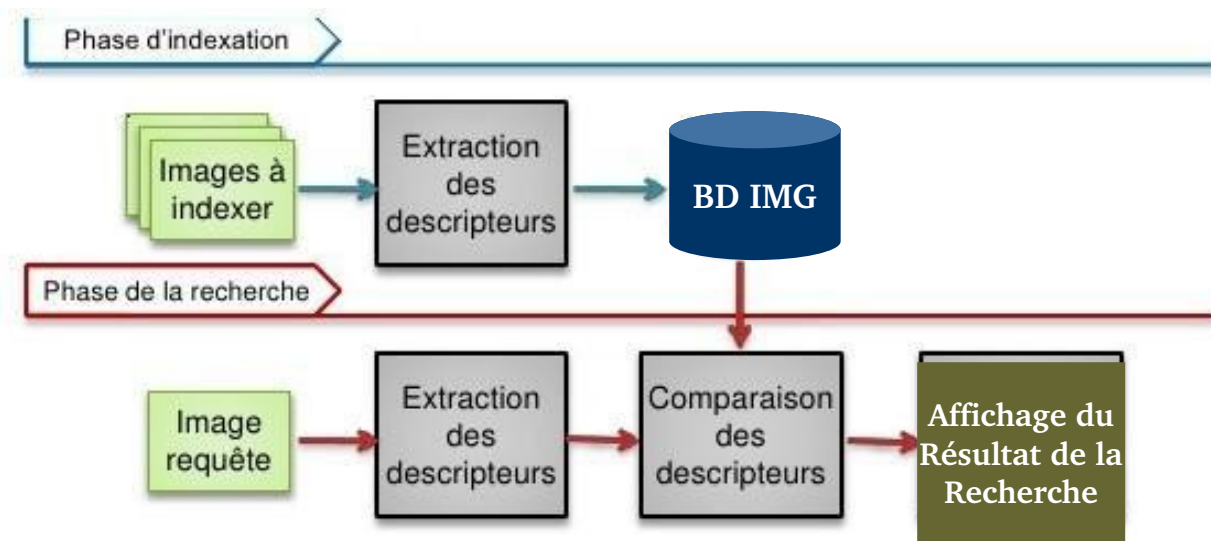
GÉNÉRALITÉ

I - Approche de Conception

Généralement, un système de recherche d'images comporte les deux principales phases suivantes:

- Une phase d'indexation
- Une phase de recherche

Pour le système implémenté, notre démarche de conception peut être se résumer par le schéma standard suivant:



Comme décrits dans le schéma de la conception, les systèmes de recherche d'images fonctionnent en général en deux phases. La phase d'indexation et la phase de recherche. La phase de recherche doit normalement être la plus rapide possible, tandis que la phase d'indexation peut-être plus longue. Cependant, lorsque l'on a un grand volume de données comme le web, elle se doit quand même d'être rapide. Le grand volume d'images à traiter impose donc que l'on extrait des descripteurs visuels rapides à calculer. La première partie de notre travail consistait à utiliser des descripteurs globaux pour trouver des correspondances entre les paires d'images. Dans les lignes suivantes, nous présentons les différents descripteurs globaux utilisés dans cette première partie.

II - Descripteurs Globaux sur les images

Une description globale est une représentation de l'image dans son ensemble, sous la forme d'un vecteur de taille fixe. Elle se contente le plus souvent d'exploiter un indice visuel unique. En indexation d'images, cette apparence visuelle globale est calculée sur toute l'image peut-être résumée par trois attributs constituant ainsi ses descripteurs. Ces trois attributs sont:

- ◆ La couleur
- ◆ La texture
- ◆ La forme

Nous les présentons brièvement ci-dessous

1 - Descripteurs Globaux sur les images – La couleur

La couleur est l'un des éléments les plus représentatifs du contenu d'une image. La prise en compte de la couleur des images a été historiquement l'une des premières caractéristiques employées pour la recherche d'images par le contenu et produit encore des résultats parfois spectaculaires sur certaines bases. Pour utiliser ou du moins extraire cette caractéristique globale de l'image, la couleur, on se réfère souvent à calculer l'*histogramme*, les *moments de hu*.

A - Histogramme

Les travaux relatifs à l'indexation de la couleur constituent la référence et marquent même les débuts de l'indexation d'images. De nombreux espaces colorimétriques existent : RVB est le plus classique mais des espaces comme CIELab ou CIELuv s'avèrent mieux adaptés à la perception humaine de la couleur et donc à la recherche d'images par similarité visuelle. La distribution des couleurs dans l'image est le plus souvent décrite par un *histogramme* couleur qui représente la fréquence d'apparition des différentes couleurs. Un tel descripteur est robuste aux principales transformations géométriques que peut subir l'image, ce qui le rend bien adapté à une recherche approximative globale.

L'histogramme ainsi représente la distribution globale des couleurs dans l'image. Son calcul consiste en une quantification de l'espace couleur choisi (RVB par exemple) suivie du calcul de l'histogramme des pixels ainsi transformés. Par exemple, si l'on considère une image RVB classique et que l'on quantifie chaque plan couleur sur 4 bins, l'histogramme résultant aura une dimension $4^3=64$. Si l'image est originellement codée sur 24 bits (la valeur de chaque plan est dans l'intervalle [0 - 255], l'histogramme couleur sur 64 bins pourrait être représenté par un «cube» 4x4x4. En pratique, ce descripteur est généralement représenté par un vecteur mono dimensionnel à 64 dimensions.

B- Moments de HU

Dans le traitement d'image, la vision par ordinateur et les champs connexes, un instant d'image ou moment est une certaine moyenne pondérée des intensités des pixels d'image, ou une fonction de ces moments, généralement choisis pour avoir une propriété attrayante ou une interprétation.

Les moments d'image sont utiles pour décrire les objets après la segmentation. Les propriétés simples de l'image qui se trouvent via les moments de l'image incluent la surface (ou l'intensité totale), son centroïde et des informations sur son orientation.

2 - Texture

La texture est la répétition d'éléments de base construits à partir de pixels qui respectent un certain ordre. La texture de l'image fournit une bonne information sur l'arrangement structurel des surfaces dans l'image. Deux grandes classes d'approches existent, l'une est basée sur des mesures statistiques alors que l'autre implique des mesures fréquentielles. L'approche statistique répandue est basée sur une analyse psycho-visuelle de la texture particulièrement bien adaptée pour l'indexation d'images. Au niveau sémantique, les textures peuvent apporter une information précieuse. En effet, elles permettent de différencier des parties d'images dont les descripteurs de couleurs sont identiques. Par exemple, le ciel (texture unie ou nuageuse) et la mer (texture des vagues).

3 - Forme

La description de la forme ou de la structure de l'image requiert en général deux niveaux de traitement. Elle nécessite en premier lieu une segmentation de l'image en régions ou au moins une extraction des contours. Ensuite, outre l'utilisation de caractéristiques simples comme la longueur ou la courbure des formes extraites, une technique simple consiste à caractériser la distribution de l'orientation des gradients sur les contours, sous la forme d'un histogramme par exemple.

D'autres approches plus sophistiquées existent, citons par exemple le codage par chaînes (codage de Freeman, codage différentiel), les descripteurs de Fourier, les moments de Hu ou de Zernike.

Les trois classes de descripteurs globaux qui viennent d'être présentées étant complémentaires, elles peuvent être combinées selon les besoins. Dans la mesure où elles caractérisent l'image dans sa globalité, ces approches permettent une recherche de l'image que l'on qualifie d'approximative.

3 - Base d'images utilisées

Toutes nos expériences se feront sur la base de données Coil-100. Elle contient 100 différents objets avec 20 images pour chaque objet donc environ deux mille images.

III - IMPLÉMENTATION

Dans la première partie de l'UE, nous avons utilisé des descripteurs globaux (moments de Hu, histogramme couleur) pour trouver des correspondances entre paires d'images. Un inconvénient de cette méthode est sa complexité en temps de calculs (quadratique en le nombre d'images). Dans ce projet, nous allons essayer d'aller au-delà en proposant un outil de recherche d'image avec une phase d'indexation. L'objectif est de trouver des images correspondant à notre image de requête dans une grande base de données d'images. Pour cela, nous allons développer un projet en différentes étapes.

1. Description Globale Du Contenu Des Images

Pour pouvoir comparer des descripteurs visuels, il faut prendre en compte le fait que les régions visuelles considérées n'ont pas forcément la même taille. Il faut donc que la représentation des descripteurs visuels obtenue soit discriminante dans le sens où elle doit permettre de bien différencier les images différentes. La représentation doit être aussi invariante dans la mesure où les régions de tailles différentes ou prises avec des luminosités différentes doivent avoir des représentations très proches.

Alors, une représentation pratique à tout cela est l'histogramme et les moments de Hu. Ainsi, dans cette première partie, nous avons utilisé les descripteurs présentés ci-dessus à savoir les histogrammes couleurs quantifiés et les moments de Hu.

2. Histogrammes couleurs quantifiés

Un histogramme peut être vu comme le nombre d'apparitions d'un élément dans un ensemble. Ainsi pour construire un histogramme de descripteurs de couleurs, nous avons utilisé deux phases: d'abord, les couleurs sont quantifiées, puis les couleurs quantifiées sont dénombrées. Chaque composante du vecteur (appelée en anglais *bin*) nous donne la quantité d'une couleur présente dans l'image. Afin de rendre les histogrammes invariants à la taille de l'image ou des régions, nous avons procédé à la normalisation de l'histogramme par le nombre de pixels. Pour prendre en compte l'information spatiale dans l'histogramme, nous avons ajouté la position

des régions. Pour calculer l'histogramme de couleur, nous calculons d'abord trois histogrammes, un pour chacune des couleurs RGB. Cela conduit à 3 histogrammes de 256 valeurs chacune. Cependant, un histogramme de 256 valeurs c'est beaucoup trop pour la comparaison des couleurs, alors nous avons donc réduit l'histogramme en 3 couleurs de M valeurs pour chacune (3 valeurs de x M au total). La réduction est simplement effectuée par une forte quantification de l'histogramme. Les valeurs couramment utilisées de M sont 16, 24 ou 32.

Pour réduire un histogramme de 256 à M valeurs, nous avons coupé l'histogramme en morceaux de 256/M valeurs, et en ajoutant des valeurs pour chaque pièce pour obtenir l'histogramme réduit. Pour simplifier la représentation et les calculs, nous avons représenté cet histogramme réduit par une seule table (au lieu de trois). Par exemple, pour M = 24, il s'agit d'un tableau de 72 valeurs, où les 24 premiers représentent les valeurs pour la couleur rouge, les 24 suivants les valeurs pour le Vert et les 24 dernières valeurs pour le Bleu.

Dans cette application pratique, pour simplifier les calculs, nous avons utilisés l'espace RVB mais soyez conscient que les autres espaces de couleurs peuvent donner de meilleurs résultats de comparaison entre les images.

3. Intersection de deux histogrammes

Pour chaque image, il y a un histogramme dont la taille a été réduite à 3 x M, ce qui est le premier calculé. L'étape suivante consiste à mesurer la similitude entre 2 images en appliquant une distance de Histogramme. Pour cela, nous vous suggérons de tester 2 scénarios et de les comparer. Le premier scénario est basé sur le calcul de l'intersection entre 2 histogrammes pour deux images. Pour ce faire, nous utiliserons la formule suivante:

$$\frac{\sum_{i=0}^{3 \times M} |Histol(i) - Histo2(i)|}{\sum_{j=0}^{3 \times M} Histol(j)}$$

4. Les moments de Hu

La troisième caractéristique utilisée pour la comparaison entre les images est le Hu Moments. Pour chaque image, nous calculons un résumé de leur Forme la représentation en utilisant Hu Moments. Pour cela, nous considérerons le moment Hu calculé sous une représentation en gris de l'image. Donc, il y aura 3 étapes pour ce calcul.

a - Conversion de l'image en degrés de gris

Dans cette étape, nous avons calculé les moments Hu uniquement sur l'image en niveaux de gris pour simplifier la tâche. Pour convertir l'échelle de gris, les images utilisent simplement (pour chaque pixel) la fonction suivante: Gris = (rouge + vert + bleu) / 3.

b - Réduction de l'image en niveaux de gris

En ce qui concerne les couleurs et les textures, il est inutile de garder tous les niveaux de gris dans le calcul de la texture. Alors, nous avons donc réduit la quantification de l'image pour passer de 256 niveaux de gris à T niveaux de gris avec T (T = 8, 16 ou 24). Cette valeur de T est aussi passée en argument au programme. Le procédé est simple, afin de réduire le nombre de niveaux de gris, nous divisons simplement chaque pixel de l'image par T.

c - Calculs des moments de Hu

d - Calcul de la distance entre deux images différentes basées sur les moments de Hu

Pour faire ça, nous utiliserons la formule simple suivante0:

$$Dist(im1,im2) = \frac{\sqrt{\sum_{i=1}^7 (param_i(im1) - param_i(im2))^2}}{7}$$

Où $param_i$ est une option (HM i) et $param(im i)$ est le paramètre sur l'image i. Le résultat pour chaque paramètre est une distance entre 0 et 1

5. Système de Recherche d'images

Pour rechercher les images les plus similaires à une image-exemple ou pour les regrouper il faut pouvoir mesurer la similarité (ou la dissimilarité) des images. Idéalement, les mesures doivent être capable de mesurer la similarité sémantique de deux images, dans la pratique elles ne sont capables que de mesurer la similarité visuelle. Il existe un grand nombre de mesures de similarité. Certaines sont des distances, c'est-à-dire des mesures qui ont les propriétés de non-négativité, réflexivité, symétrie et qui respectent l'inégalité triangulaire. Certaines mesures sont spécifiques aux histogrammes ou aux distributions. Alors, notre système de recherche d'images s'appuie sur les caractéristiques que nous avons calculées et présentées précédemment.

Pour cette implémentation, tout se fait en ligne de commande et la sortie est affichée sous forme de texte dans la console.

En argument (entrée) du programme, il faut spécifier le nom d'un fichier image. Ensuite, le système calcule la distance entre cette image et toutes les images de la base d'images. En sortie, le système affiche les N premières images, c'est-à-dire triées dans l'ordre de distance de la plus petite à la plus grande. Un argument d'entrée permet de spécifier N (par défaut 10). Pour cela, nous avons utilisé l'algorithme des k plus proches voisins.

6. Fonction Globale de Similarité entre deux images

Notre système utilise deux descripteurs différents à savoir l'histogramme de couleur et les moments de Hu, alors nous avons donc deux distances: celle issue du descripteur couleur et celle issue des moments de Hu. Afin de pouvoir combiner ces deux descripteurs, nous définissons une fonction globale de similarité entre deux images qui consiste en la somme pondérée de ces deux distances :

$$\text{Distance(globale)} = \text{poids} * \text{Distance(couleur)} + (1 - \text{poids}) * \text{Distance(momentsHu)}$$

Avec poids prend une valeur entre 0 et 1 (à passer en argument de votre programme avec une valeur par défaut égale à 0.5).

7. Clustering

Afin d'accélérer les performances de notre système, nous avons intégré un algorithme de clustering qui nous a permis d'accélérer la phase de recherche. L'une des méthodes les plus simples utilisées pour la création de cluster est le k-means.

III - EXPÉRIENCE ET RÉSULTATS

Ici, nous allons présenter les résultats des expériences faites sur nos implémentations. Dans tous les cas, on sépare les images en 70% d'apprentissage et en 30% de test.

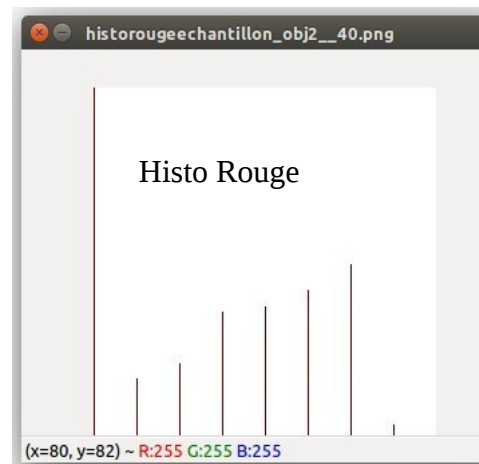
1- Description Globale des Contenus - Résultats

Dans la première partie, nous avons utilisé des descripteurs globaux tels que les moments de Hu et les histogrammes couleurs pour trouver des correspondances entre paires d'images, nous présentons les résultats ci-dessous pour les descripteurs utilisés.

Les histogrammes couleurs quantifiés - Résultats

Pour calculer l'histogramme de couleur, nous calculons d'abord trois histogrammes, un pour chacune des couleurs RGB. Cela conduit à 3 histogrammes de 256 valeurs chacune. Cependant, un histogramme de 256 valeurs c'est beaucoup trop pour la comparaison des couleurs, alors nous avons donc réduit l'histogramme en 3 couleurs de M valeurs pour chacune (3 valeurs de x M au total). On peut faire les tests avec les valeurs 8, 16, 24, par couleur. Ci-dessous, on présente un histogramme pour chacune des images avec la valeur de 8. C'est-à-dire on va avoir un histogramme de 8 échantillons. Voir ci-dessous.

Commande utilisée: `./knn objet2__40.png 8 3` (on expliquera plus tard ce fonctionnement)



Ici on présente les histogrammes échantillonnés (8) pour les 3 couleurs utilisées sur une même image. On remarque la faible apparition de la couleur bleu et une forte apparition pour le vert. C'est très logique compte tenu de la couleur originale de l'image.

Moments de Hu - Résultats

Nous avons calculés les **moments de hu d'ordre 8** sur toutes les images et les rediriger dans un vecteur et récupérer nos résultat pour faire d'autres calculs. Mais on tient d'abord à présenter une capture des différents moments obtenus.

Pour obtenir des moments dur l'objet2__40.png, on exécute la commande

`./knnhu obj2__10.png 1` (`./knn nom_objet`, on explique le paramètre 1 plu-tard).

```
gdorleon@unknown:~/Documents/MasterIFI/M2/Indexation/Projet/codes/knnhu$ ./knnhu obj2__10.png 1
test/obj2__10.png
0.00107496
6.30488e-09
7.40779e-06
2.17428e-05
2.30252e-10
-1.76361e-09
-1.5208e-10
-2.96059e-10
```

On constate clairement les 8 valeurs puisque nous, nous avons calculé un moment hu d'ordre 8

2- Systèmes de recherche d'images - Résultats

L'une des meilleures approches de notre système c'est de permettre la recherche des images en fournissant. Comme décrit plus haut et comme c'est requis dans la description du proejt , pour l'implémentation de cette fonction, tout se en ligne de commande et la sortie est affichée sous forme texte dans la console.

En argument (entrée) du programme, il faut spécifie le nom d'un fichier image.Ensuite, le système calcule la distance entre cette image et toutes les images de la base d'images. En sortie, le système affiche les N premières images, c'est-à-dire triées dans l'ordre de distance de la plus petite à la plus grande. Un argument d'entrée permet de spécifier N (par défaut 10). Pour cela, nous avons utilisés l'algorithme des k plus proches voisins.

Fonctionnement du programme

Pour compiler ce programme, veuillez se placer dans le répertoire knn, puis, ouvrant un terminal, on tape:

- 1- `cmake .`
- 2 - `make`
- 3 - `./knn image_name nb_échantillon nb_voisins`

En effet, le descripteur utilisé ici pour faire la recherche est l'histogramme de couleur quantifié. Pourquoi ce choix ?

Il existe beaucoup de descripteurs pour faire la recherche d'une image. Mais, on a choisi d'utiliser l'histogramme. L'avantage d'utiliser l'histogramme, c'est qu'il permet de détecter plus efficacement les couleurs, ce qui est mieux adapté ici à notre recherche. Le moment de Hu par exemple lui se base sur la forme, ce qui peut fausser notre résultat.

- En entrée, notre programme prend comme paramètres:

1- nom de l'image

2- nombre de niveaux pour l'échantillonnage

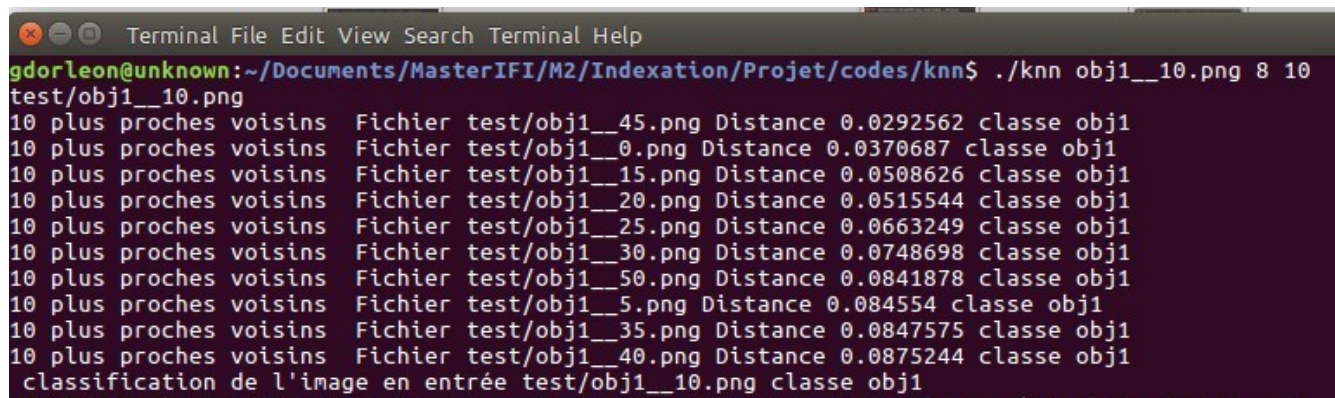
3- nombre de voisins qui est par défaut 10.

Exemple de commande: `./knn obj1__10.png 8 10`

- En sortie, notre programme de recherche d'images renvoie comme résultat a liste des k plus proches voisins ainsi que la distance par rapport à l'image passée en entrée.

Quelques résultats ci-dessous

Nous avons fait un test sur l'image `obj1__10.png` avec 10 plus proches voisins, voir suivant:



```
Terminal File Edit View Search Terminal Help
gdorleon@unknown:~/Documents/MasterIFI/M2/Indexation/Projet/codes/knn$ ./knn obj1__10.png 8 10
test/obj1__10.png
10 plus proches voisins Fichier test/obj1__45.png Distance 0.0292562 classe obj1
10 plus proches voisins Fichier test/obj1__0.png Distance 0.0370687 classe obj1
10 plus proches voisins Fichier test/obj1__15.png Distance 0.0508626 classe obj1
10 plus proches voisins Fichier test/obj1__20.png Distance 0.0515544 classe obj1
10 plus proches voisins Fichier test/obj1__25.png Distance 0.0663249 classe obj1
10 plus proches voisins Fichier test/obj1__30.png Distance 0.0748698 classe obj1
10 plus proches voisins Fichier test/obj1__50.png Distance 0.0841878 classe obj1
10 plus proches voisins Fichier test/obj1__5.png Distance 0.084554 classe obj1
10 plus proches voisins Fichier test/obj1__35.png Distance 0.0847575 classe obj1
10 plus proches voisins Fichier test/obj1__40.png Distance 0.0875244 classe obj1
classification de l'image en entrée test/obj1__10.png classe obj1
```

Notre système effectue efficacement la recherche pour l'image donnée en entrée en se basant sur l'histogramme. Le système renvoie la distance, la classe des k=10 voisins les plus proches. Essayons avec une autre image d'entrée et un autre nombre de voisins différents.

Nous essayons avec l'image `obj2__20.png` avec k=15 voisins.

Commande utilisée: `./knn obj2__20.png 8 15`

Voir le résultat ci-dessous

```

gdorleon@unknown:~/Documents/MasterIFI/M2/Indexation/Projet/codes/knn$ ./knn obj2__20.png 8 15
test/obj2__20.png
15 plus proches voisins Fichier test/obj2__15.png Distance 0.0211589 classe obj2
15 plus proches voisins Fichier test/obj2__45.png Distance 0.0324707 classe obj2
15 plus proches voisins Fichier test/obj2__0.png Distance 0.0372314 classe obj2
15 plus proches voisins Fichier test/obj2__25.png Distance 0.044637 classe obj2
15 plus proches voisins Fichier test/obj2__30.png Distance 0.0450846 classe obj2
15 plus proches voisins Fichier test/obj2__10.png Distance 0.0539551 classe obj2
15 plus proches voisins Fichier test/obj2__35.png Distance 0.0570475 classe obj2
15 plus proches voisins Fichier test/obj2__40.png Distance 0.0656738 classe obj2
15 plus proches voisins Fichier test/obj2__5.png Distance 0.0774333 classe obj2
15 plus proches voisins Fichier test/obj3__70.png Distance 0.080363 classe obj3
15 plus proches voisins Fichier test/obj3__65.png Distance 0.560466 classe obj3
15 plus proches voisins Fichier test/obj3__60.png Distance 0.577596 classe obj3
15 plus proches voisins Fichier test/obj3__55.png Distance 0.610718 classe obj3
15 plus proches voisins Fichier test/obj3__50.png Distance 0.636475 classe obj3
15 plus proches voisins Fichier test/obj3__5.png Distance 0.660319 classe obj3
classification de l'image en entrée test/obj2__20.png classe obj3
gdorleon@unknown:~/Documents/MasterIFI/M2/Indexation/Projet/codes/knn$

```

Le système renvoie efficacement les k=15 voisins les plus proches de l'image obj2__20.png. Regardons les distances, c'est exactement dans l'ordre croissant de la plus petite distance de la valeur de k la plus petite à la plus grande distance de la plus grande valeur de k.

1^{er} voisin » » : obj2__15.png , sa distance est de 0,0211589.

k15^e voisin » » : objet3__ 5.png, sa distance est de 0, 0660319.

En conclusion, on peut affirmer que notre système recherche efficacement une image donnée en entrée en se basant sur l'histogramme comme descripteur et en utilisant l'algorithme des k plus proches voisins. On peut lancer la recherche pour toutes les images de la base, ou de n'importe quelle base d'image, cela marchera parfaitement pourvu qu'on change le répertoire dans le code, à la ligne 32 (*#define RACINE "test/"*) et que la machine utilisée puisse lire une plus grande base. On procédera plus bas dans ce rapport à l'évaluation de notre système.

3- Fonction Globale de Similarité entre deux images - Résultats

Notre système utilise deux descripteurs différents à savoir l'histogramme de couleur et les moments de Hu, alors nous avons donc deux distances: celle issue du descripteur couleur et celle issue des moments de Hu. Afin de pouvoir combiner ces deux descripteurs, nous définissons une fonction globale de similarité entre deux images qui consiste en la somme pondérée de ces deux distances :

$$Distance(globale) = poids * Distance(couleur) + (1 - poids) * Distance(momentsHu)$$

Avec poids qui prend une valeur entre 0 et 1 (à passer en argument de votre programme avec une valeur par défaut égale à 0.5).

Fonctionnement du programme

Pour lancer ce programme, il faut se placer dans le répertoire knnglobal puis, ouvrant un terminal, on tape:

cmake . , puis on tape

make , puis on tape

./knnglobal nom_objet nb_echantillon nb_voisin

Pour cette fonction globale de similarité, **les descripteurs utilisés sont une combinaison de l'histogramme couleur et moment de Hu** déjà calculés plus haut.

- En entrée, notre programme prend en paramètres:

1- nom de l'image

2- nombre de niveau pour l'échantillonnage

3- nombre de voisins

4- le poids entre 0 et 1

Exemple de commande: **./knnglobal obj1__10.png 8 10 0.5**

Où: nom_image: obj1__10.png

nb_échnatillon: 8

nb_voisins: 10

le poids: 0.5

- En sortie on obtient comme résultat: la liste des k plus proches voisins ainsi que les distance par rapport à l'image passée en entrée et leur classe.

Nous présentons ci-dessous les résultats obtenus sur des images issues de 5 classes différentes en se basant sur les deux descripteurs combinés.

La sortie nous renvoie la liste des 10 plus proches voisins, leur distance ainsi que leur classe. La distance varie d'une classe, d'un voisin à un autre. Voir les captures pour les 5 classes différentes présentées ci-dessous.

Image obj1__10.png , Classe obj1

```
Terminal File Edit View Search Terminal Help
gdorleon@unknown:~/Documents/MasterIFI/M2/Indexation/Projet/codes/knnglobal$ ./knnglobal obj1__10.png 8 10 0.5
test/obj1__10.png
10 plus proches voisins Fichier test/obj1__5.png Distance 5.4589e-06 classe obj1
10 plus proches voisins Fichier test/obj1__10.png Distance 7.82894e-06 classe obj1
10 plus proches voisins Fichier test/obj1__45.png Distance 8.59155e-06 classe obj1
10 plus proches voisins Fichier test/obj1__40.png Distance 1.02294e-05 classe obj1
10 plus proches voisins Fichier test/obj1__50.png Distance 1.21604e-05 classe obj1
10 plus proches voisins Fichier test/obj1__0.png Distance 1.27985e-05 classe obj1
10 plus proches voisins Fichier test/obj1__15.png Distance 1.44681e-05 classe obj1
10 plus proches voisins Fichier test/obj1__35.png Distance 1.58684e-05 classe obj1
10 plus proches voisins Fichier test/obj1__20.png Distance 1.89728e-05 classe obj1
10 plus proches voisins Fichier test/obj1__30.png Distance 1.92801e-05 classe obj1
classification de l'image en entrée test/obj1__10.png classe obj1
gdorleon@unknown:~/Documents/MasterIFI/M2/Indexation/Projet/codes/knnglobal$
```


Image obj2__ 5. png , Classe Obj2

```
Terminal File Edit View Search Terminal Help
classification de l'image en entrée test/obj1__10.png classe obj1
gdorLeon@unknown:~/Documents/MasterIFI/M2/Indexation/Projet/codes/knnglobal$ ./knnglobal obj2__5.png 8 10 0.5
test/obj2__5.png
10 plus proches voisins Fichier test/obj2__15.png Distance 8.44607e-08 classe obj2
10 plus proches voisins Fichier test/obj2__35.png Distance 4.63162e-07 classe obj2
10 plus proches voisins Fichier test/obj2__30.png Distance 5.0399e-07 classe obj2
10 plus proches voisins Fichier test/obj2__0.png Distance 6.84855e-07 classe obj2
10 plus proches voisins Fichier test/obj2__20.png Distance 1.32521e-06 classe obj2
10 plus proches voisins Fichier test/obj2__25.png Distance 2.25115e-06 classe obj2
10 plus proches voisins Fichier test/obj2__45.png Distance 3.30513e-06 classe obj2
10 plus proches voisins Fichier test/obj2__10.png Distance 4.21636e-06 classe obj2
10 plus proches voisins Fichier test/obj2__5.png Distance 4.87866e-06 classe obj2
10 plus proches voisins Fichier test/obj2__40.png Distance 5.04814e-06 classe obj2
classification de l'image en entrée test/obj2__5.png classe obj2
gdorLeon@unknown:~/Documents/MasterIFI/M2/Indexation/Projet/codes/knnglobal$
```

Image obj3__ 5. png

```
Terminal File Edit View Search Terminal Help
gdorLeon@unknown:~/Documents/MasterIFI/M2/Indexation/Projet/codes/knnglobal$ ./knnglobal obj3__5.png 8 10 0.5
test/obj3__5.png
10 plus proches voisins Fichier test/obj3__0.png Distance 8.6411e-07 classe obj3
10 plus proches voisins Fichier test/obj5__25.png Distance 3.26076e-06 classe obj5
10 plus proches voisins Fichier test/obj5__20.png Distance 3.31204e-06 classe obj5
10 plus proches voisins Fichier test/obj5__15.png Distance 3.39943e-06 classe obj5
10 plus proches voisins Fichier test/obj5__30.png Distance 3.71897e-06 classe obj5
10 plus proches voisins Fichier test/obj5__0.png Distance 4.57496e-06 classe obj5
10 plus proches voisins Fichier test/obj3__10.png Distance 4.60449e-06 classe obj3
10 plus proches voisins Fichier test/obj5__40.png Distance 6.20387e-06 classe obj5
10 plus proches voisins Fichier test/obj1__60.png Distance 6.85612e-06 classe obj1
10 plus proches voisins Fichier test/obj4__355.png Distance 7.02655e-06 classe obj4
classification de l'image en entrée test/obj3__5.png classe obj5
gdorLeon@unknown:~/Documents/MasterIFI/M2/Indexation/Projet/codes/knnglobal$
```

Avec l'image objet3__5.png, on constate alors que d'autres images des autres classes appartiennent aux voisins de l'objet3__5.png. Cela s'explique par le fait du nombre d'images contenues dans notre base test pour la classe obj3, dans la base test, il ya moins que 10 images. On ne peut absolument pas faire le test sur toutes les images de la base Coil_100 car le temps de calcul est trop long et dépasse la capacité de notre machine.

Image obj4__295.png

```
Terminal File Edit View Search Terminal Help
gdorLeon@unknown:~/Documents/MasterIFI/M2/Indexation/Projet/codes/knnglobal$ ./knnglobal obj4__295.png 8 10 0.5
test/obj4__295.png
10 plus proches voisins Fichier test/obj4__305.png Distance 7.76167e-06 classe obj4
10 plus proches voisins Fichier test/obj4__310.png Distance 1.04492e-05 classe obj4
10 plus proches voisins Fichier test/obj1__40.png Distance 1.34534e-05 classe obj1
10 plus proches voisins Fichier test/obj4__315.png Distance 1.53225e-05 classe obj4
10 plus proches voisins Fichier test/obj1__45.png Distance 1.53472e-05 classe obj1
10 plus proches voisins Fichier test/obj4__320.png Distance 1.56176e-05 classe obj4
10 plus proches voisins Fichier test/obj1__35.png Distance 1.69282e-05 classe obj1
10 plus proches voisins Fichier test/obj1__30.png Distance 1.798e-05 classe obj1
10 plus proches voisins Fichier test/obj1__50.png Distance 1.93431e-05 classe obj1
10 plus proches voisins Fichier test/obj4__340.png Distance 1.9954e-05 classe obj4
classification de l'image en entrée test/obj4__295.png classe obj4
gdorLeon@unknown:~/Documents/MasterIFI/M2/Indexation/Projet/codes/knnglobal$
```

Image obj5__5.png

```
Terminal File Edit View Search Terminal Help
gdorLeon@unknown:~/Documents/MasterIFI/M2/Indexation/Projet/codes/knnglobal$ ./knnglobal obj5__5.png 8 10 0.5
test/obj5__5.png
10 plus proches voisins Fichier test/obj5__0.png Distance 8.44884e-07 classe obj5
10 plus proches voisins Fichier test/obj5__15.png Distance 1.65794e-06 classe obj5
10 plus proches voisins Fichier test/obj5__20.png Distance 2.95929e-06 classe obj5
10 plus proches voisins Fichier test/obj5__25.png Distance 5.12248e-06 classe obj5
10 plus proches voisins Fichier test/obj1__55.png Distance 5.42059e-06 classe obj1
10 plus proches voisins Fichier test/obj3__0.png Distance 5.81735e-06 classe obj3
10 plus proches voisins Fichier test/obj3__5.png Distance 6.20387e-06 classe obj3
10 plus proches voisins Fichier test/obj5__30.png Distance 6.88998e-06 classe obj5
10 plus proches voisins Fichier test/obj5__40.png Distance 7.22879e-06 classe obj5
10 plus proches voisins Fichier test/obj1__60.png Distance 8.13084e-06 classe obj1
classification de l'image en entrée test/obj5__5.png classe obj5
gdorLeon@unknown:~/Documents/MasterIFI/M2/Indexation/Projet/codes/knnglobal$
```

Même constat pour l'image objet 3 et obj4, nous avons des images des autres classes pour l'objet5 car dans notre base, ces classes ont moins de 10 images.

On peut sans problème augmenter ou utiliser toute la base, cela ne va déranger le programme en aucun cas, tout fonctionnera très bien.

Un autre constat à signaler c'est que, en comparaison avec la recherche qui se basait uniquement sur l'histogramme de couleur. C'est que, avec la même image, on trouve le même résultat pour les voisins mais les distances sont différentes.

Clustering – Résultats

Afin d'accélérer les performances de notre système, nous avons intégré un algorithme de clustering qui nous a permis d'accélérer la phase de recherche. L'une des méthodes les plus simples utilisés pour la création de cluster est le k-means.

Ainsi, pour faire le clustering, nous avons utilisé l'algorithme du k-means ci-dessous:

- 1 - Initialisation en choisissant un nombre défini k et un nombre de clusters.
- 2 - Affectation de tous les points aux centroïde le plus proche.
- 3 - Déclaration de chaque centre de gravité (moyenne) des points assignés. Dans le cas où la moyenne n'a pas de points attribués, on choisit une nouvelle position pour la moyenne.

Et on répète les étapes 2 et 3 jusqu'à ce que les affectations des à l'étape 2 ne changent pas.

Fonctionnement du programme

Pour lancer ce programme, il faut se placer dans le répertoire kmean puis, ouvrant un terminal, on tape:

cmake . , puis on tape

make , puis on tape

./kmean 8 3

Pour implémenter notre algorithme de *kmeans*, nous avons utilisé l'histogramme couleur comme descripteur.

- En entrée, notre programme prend 2 paramètres:
 - 1 - nombre de niveau pour l'échantillonnage
 - 2 - le nombre k de cluster
- En sortie, nous obtenons comme résultat la liste d'image de la base d'image avec le cluster associé.

Résultats avec k-means

Commande utilisée: `./kmean 8 3`

Avec cette commande, on demande à notre système de classer notre base en 3 classes d'images en se basant sur l'algorithme k-means. En voici notre résultat propre et net des 3 classes ainsi que leurs contenu.

```
Terminal File Edit View Search Terminal Help
gdorleon@unknown:~/Documents/MasterIFI/M2/Indexation/Projet/codes/kmean$ ./kmean 8 3
image : test/obj10__185.png cluster numéro 0
image : test/obj10__190.png cluster numéro 0
image : test/obj10__195.png cluster numéro 0
image : test/obj2__105.png cluster numéro 1
image : test/obj2__110.png cluster numéro 1
image : test/obj2__115.png cluster numéro 1
image : test/obj2__120.png cluster numéro 1
image : test/obj1__0.png cluster numéro 2
image : test/obj1__10.png cluster numéro 2
image : test/obj1__15.png cluster numéro 2
image : test/obj1__5.png cluster numéro 2
image : test/obj3__0.png cluster numéro 2
image : test/obj3__10.png cluster numéro 2
image : test/obj3__5.png cluster numéro 2
```

Commande utilisée: `./kmean 16 4`

On fait varier un peu notre requête, cette fois ci on utilise un échantillonnage à 16 niveaux et on demande au système de nous faire un résultat en 4 clusters. Tout est bien fait, voir ci-dessous.

```
gdorleon@unknown:~/Documents/MasterIFI/M2/Indexation/Projet/codes/kmean$ ./kmean 16 4
image : test/obj10__185.png cluster numéro 0
image : test/obj2__105.png cluster numéro 1
image : test/obj2__110.png cluster numéro 1
image : test/obj2__115.png cluster numéro 1
image : test/obj2__120.png cluster numéro 1
image : test/obj10__190.png cluster numéro 2
image : test/obj10__195.png cluster numéro 2
image : test/obj1__0.png cluster numéro 3
image : test/obj1__10.png cluster numéro 3
image : test/obj1__15.png cluster numéro 3
image : test/obj1__5.png cluster numéro 3
image : test/obj3__0.png cluster numéro 3
image : test/obj3__10.png cluster numéro 3
image : test/obj3__5.png cluster numéro 3
gdorleon@unknown:~/Documents/MasterIFI/M2/Indexation/Projet/codes/kmean$
```

IV- CONCLUSION DE LA PREMIÈRE PARTIE

Dans cette première partie, nous avons utilisés les descripteurs globaux sur les images.

Nous avons présenter l'implémentation et les résultats de la description globale des contenus d'une image avec des différents descripteurs utilisés comme les moments de hu et l'histogramme de couleur quantifié.

Dans une deuxième phase, nous avons implémenter un système de recherche d'images par l'implémentation de l'algorithme des k plus proches voisins et l'utilisation de l'histogramme comme descripteur.

Ensuite, dans une troisième phase, nous avons implémenter un système à partir d'une fonction globale de similarité entre les images en combinant les deux descripteurs présentés, histogramme de couleur quantifié et moment de hu et en utilisant un poids.

Dans une quatrième partie, nous avons fait la classification des objets en utilisant l'algorithme de k-means afin d'accélérer la performance de notre système.

Et enfin, nous avons implémenté deux fonctions permettent de faire l'évaluation de notre système, le rappel et la précision qui sont présentés à la fin du rapport.

Dans la suite du document, nous attaquons les descripteurs locaux ainsi que l'approche du bag-of-words.

- Deuxième Partie -

Utilisations des Descripteurs Locaux et Approche Bag-of-Words

I - GÉNÉRALITÉ

A l'opposé des approches globales présentées dans la première partie, les approches des descripteurs locaux visent à décrire le contenu de l'image localement. Elles offrent ainsi la possibilité d'effectuer une recherche sur une partie de l'image ou encore sur un objet présent dans l'image. La sélection étant définie explicitement sur une partie de l'image, on a l'habitude de leur associer le paradigme de requêtes partielles. Pour les bases d'images génériques, deux techniques de description entrent dans ce cadre. Elles mettent en jeu des primitives de l'image de nature différente:

1 - Des régions d'intérêt

La phase d'indexation de l'image consiste à découper celle-ci en régions selon une approche de segmentation. Lorsque l'on n'a aucune connaissance a priori sur le contenu de la base d'images, il est difficile de réaliser une segmentation qui corresponde précisément aux objets que l'on souhaitera reconnaître par la suite. En général, il est préférable que la segmentation réalisée reste assez grossière, de façon à produire un petit nombre de régions qui constituent des zones visuellement cohérentes et pertinentes comme requête. Pour être discriminantes, ces régions doivent présenter une certaine variabilité disons photométrique, qu'il faut décrire finement.

II - IMPLÉMENTATION

Pour cette deuxième partie de notre travail, nous allons construire un système parallèle au premier et pour ensuite comparer les résultats. Cette seconde approche repose sur l'utilisation de descripteurs locaux à partir d'un processus en 2 étapes:

1. Une partie apprentissage comprenant 2/3 de toutes les données.
2. Une partie recherche où vous allez comparer une image requête au résultat de la phase d'apprentissage.

1. PHASE D' APPRENTISSAGE

Le but de cette phase est de construire un dictionnaire qui sera utilisé pour appliquer la technique des Bag-of-Visual-Words. Afin d'atteindre cet objectif, nous allons implémenter les fonctionnalités suivantes.

a - Détection des points d'intérêts

Les points d'intérêt (ou point of interest) sont définis comme étant des points qui possèdent des caractéristiques qui permettent de les distinguer des autres points de l'image.

b - Calcul d'un descripteur local pour chacun des keypoints

Le descripteur le plus utilisé actuellement est de descripteur SIFT. Il vous faudra cependant comprendre comment récupérer ce descripteur pour pouvoir calculer la prochaine étape: le clustering des keypoints en mots-visuels.

2 - Définition du dictionnaire de mots visuels, BOW

Le modèle de sac de mots (modèle BoW) peut être appliqué à la classification d'image, en traitant les fonctions d'image en tant que mots. Dans la classification des documents, un sac de mots est un vecteur épars de recensement d'occurrences de mots; C'est-à-dire un histogramme discret sur le vocabulaire. En d'autres termes, un sac de mots visuels est un vecteur de comptes d'occurrence d'un vocabulaire des caractéristiques d'image locales.

a - Implémentation du BOW

Dans cette partie, nous allons implémenter l'algorithme du sac de mots visuels (BOW).

La première étape du modèle Bag-of-words consiste à utiliser l'algorithme SIFT pour extraire les vocabulaires visuels de chaque type d'image et rassembler tous les vocabulaires visuels.

La deuxième étape consiste à construire la liste des mots à l'aide de l'algorithme K-Means. L'algorithme de K-Means est une mesure de similarité entre les échantillons basée sur la méthode de clustering indirect; cet algorithme avec K comme paramètre, des N objets sont divisés en K clusters, de manière à avoir un degré élevé de similarité au sein d'un cluster, inter-cluster et avec une similarité est faible. Le SIFT alors extrait le vocabulaire visuel entre la distance en fonction de la distance.

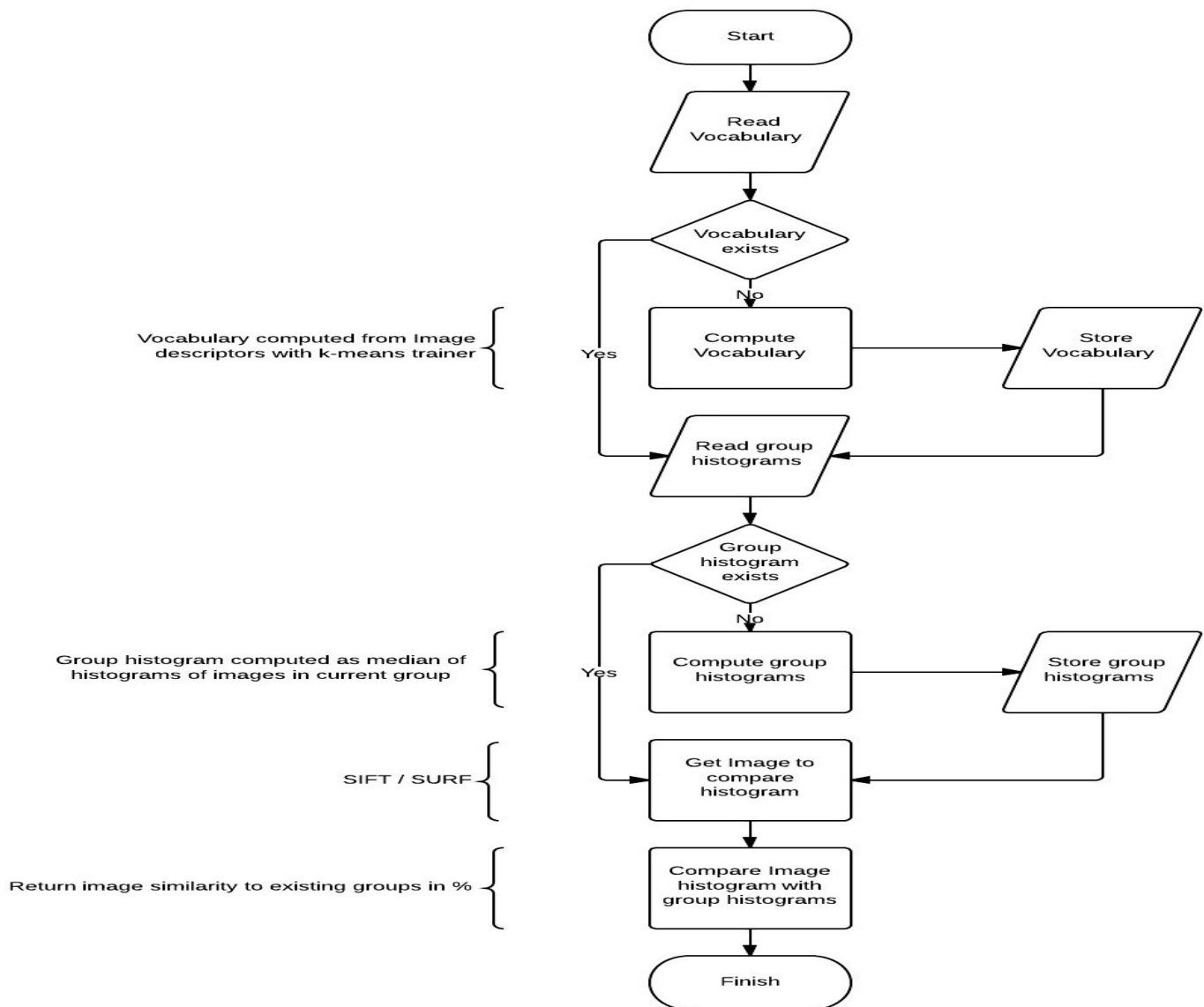
La troisième étape consiste à utiliser le vocabulaire du tableau des mots pour représenter l'image. En utilisant l'algorithme SIFT, vous pouvez extraire plusieurs points de fonctionnalité de chaque image. Ces points caractéristiques peuvent être remplacés par des mots dans la liste des mots. En comptant le nombre de mots dans chaque mot dans la liste des mots, l'image peut s'exprimer comme en K dimensions.

L'algorithme du BOW agit de la façon suivante:

- Calculer le vocabulaire des mots visuels avec l'algorithme k-means (où k est équivalent avec le nombre de mots visuels dans le vocabulaire).
- Calculer les histogrammes normalisés de groupe pour de meilleurs résultats. Cette partie nécessite un vocabulaire calculé. L'histogramme de groupe est un histogramme normalisé, cela signifie que la somme de toutes les colonnes dans l'histogramme est égale à 1.
- Calculer l'histogramme pour l'image sur l'entrée et la comparer avec tous les histogrammes de groupe pour réaliser à quelle image de groupe appartient. Ceci a été implémenté comme une intersection d'histogramme.

Sur la figure suivante, on présente le schéma de l'implémentation de l'algorithme de BOW pour le calcul du vocabulaire et du histogramme de groupe.

Voir image ci-dessous:



Fonctionnement du programme

Pour lancer ce programme, il faut se placer dans le répertoire SIFTBOW puis, ouvrant un terminal, on tape:

python SIFTBOW.py

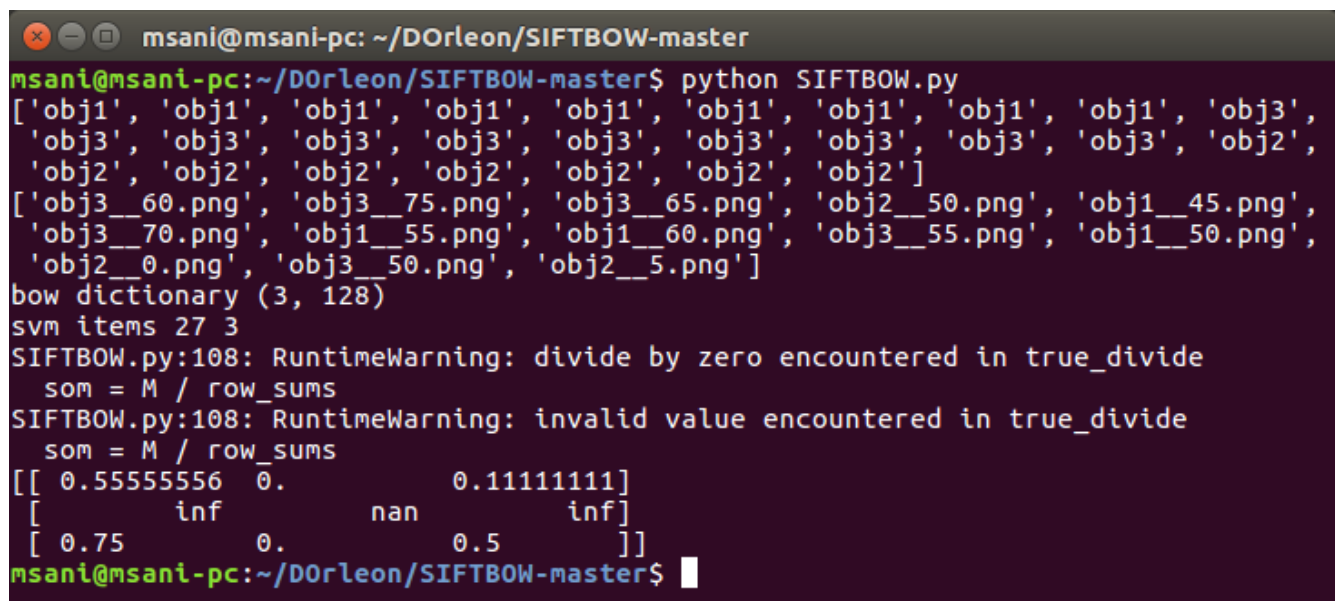
- En entrée, notre programme prend comme argument un ensemble d'apprentissage et un ensemble de test.

L'ensemble d'entraînement contient des sous-dossiers qui représentent les différentes classes.

- En sortie, notre système renvoie une matrice de confusion qui indique les performances du système, le dictionnaire des mots et le nombre d'objets et le nombre de classes.

III- EXPÉRIENCE ET RÉSULTATS

La capture d'écran ci-dessous donne un aperçu du résultat de la sortie du sac de mots visuels sur notre base de test.



```
msani@msani-pc: ~/DOrleon/SIFTBOW-master
msani@msani-pc:~/DOrleon/SIFTBOW-master$ python SIFTBOW.py
['obj1', 'obj1', 'obj1', 'obj1', 'obj1', 'obj1', 'obj1', 'obj1', 'obj1', 'obj1', 'obj3',
 'obj3', 'obj3', 'obj3', 'obj3', 'obj3', 'obj3', 'obj3', 'obj3', 'obj3', 'obj2',
 'obj2', 'obj2', 'obj2', 'obj2', 'obj2', 'obj2', 'obj2']
['obj3__60.png', 'obj3__75.png', 'obj3__65.png', 'obj2__50.png', 'obj1__45.png',
 'obj3__70.png', 'obj1__55.png', 'obj1__60.png', 'obj3__55.png', 'obj1__50.png',
 'obj2__0.png', 'obj3__50.png', 'obj2__5.png']
bow dictionary (3, 128)
svm items 27 3
SIFTBOW.py:108: RuntimeWarning: divide by zero encountered in true_divide
  som = M / row_sums
SIFTBOW.py:108: RuntimeWarning: invalid value encountered in true_divide
  som = M / row_sums
[[ 0.55555556  0.          0.11111111]
 [          inf          nan          inf]
 [ 0.75       0.          0.5       ]]
msani@msani-pc:~/DOrleon/SIFTBOW-master$
```

a) Interprétation du tableau des résultats

Le tableau affiche d'abord le nom des objets résultants de la base test ensuite la matrice de confusion, le dictionnaire des mots, le nombre de mots visuels k dans le vocabulaire.

Alors, dans ce tableau de résultat, on a de façon détaillée:

bow dictionary (3, 128) représente le dictionnaire de mots pour nos 3 classes.

Svm(27, 3) donne le nombre d'objets contenus dans nos 3 classes, ici 27.

Dans la matrice de confusion:

La ligne [0,0] signifie que 55% des objets de la classe 1 sont correctement prédites c'est-à-dire le nombre d'objets bien classés sur le nombre d'objets contenus dans la classe 1.

Pour la classe 2, la valeur «nan» signifie qu'il n'y a aucun objet dans la base test pour cette classe.

Les éléments de la classe 3, 3^e ligne de la matrice, 50% sont bien classés. Par contre 75% des objets de la classe 1 sont attribués à la classe 3, (colonne[0,1]). 11% des éléments de la classe 1 sont attribués à la classe 3. Et, comme il n'y a pas d'objets présents dans la base de test pour la classe 2, on voit alors les 0 qui traduisent qu'aucun élément de la classe 2 n'a été attribué à aucune autre classe.

IV - CONCLUSION BAGS-OF-VISUAL-WORDS

En fait on constate que le BOW présente de nombreux avantages et inconvénients. Par exemple, on peut citer comme avantage que le BOW est en grande partie non affecté par la position et l'orientation de l'objet dans image. Un vecteur de longueur fixe indépendamment du nombre de détections. Le BOW est très réussie dans la classification des images selon l'objets qu'ils contiennent mais a encore besoin de tests supplémentaires pour les grands changements d'échelle et point de vue.

Comme désavantages, on constate qu'aucune utilisation explicite de la configuration des positions visuelles et que le BOW peut être mauvais lors de la localisation d'objets dans une image.

I. ÉVALUATION

Pour évaluer la performance de notre système, deux méthodes sont utilisées: Le rappel et la précision.

1 - Rappel

Considérant notre base de données, le rappel est alors le nombre de documents pertinents trouvés au regard du nombre de documents pertinents trouvés dans la base. Cela s'explique comment ? Chaque fois qu'un utilisateur interroge la base, il souhaite alors recevoir les documents qui correspondent à ses besoins. Ainsi, si cette concordance entre la requête de l'utilisateur et le résultats trouvés est importante, alors le taux de rappel est élevé. À l'inverse, si dans la liste de réponse, aucun document pertinent n'est retrouvé en dépit du nombre d'importance de document dans la base, on parlera de silence, l'inverse du rappel.

Nous utilisons une formule simple pour calculer le rappel du système:

$$\text{rappel}_i = \frac{\text{nb de documents correctement attribués à la classe } i}{\text{nb de documents appartenant à la classe } i}$$

2- Précision

C'est le nombre de documents pertinents renvoyés par la base par rapport au nombre de document total proposé par un moteur de recherche pour une requête donnée. Le taux de précision se calcule par la formule suivante:

$$\text{précision}_i = \frac{\text{nb de documents correctement attribués à la classe } i}{\text{nb de documents attribués à la classe } i}$$

Fonctionnement du programme

Pour lancer ce programme, il faut se placer dans le répertoire knnlearn puis, ouvrant un terminal, on tape:

cmake . , puis on tape

make , puis on tape

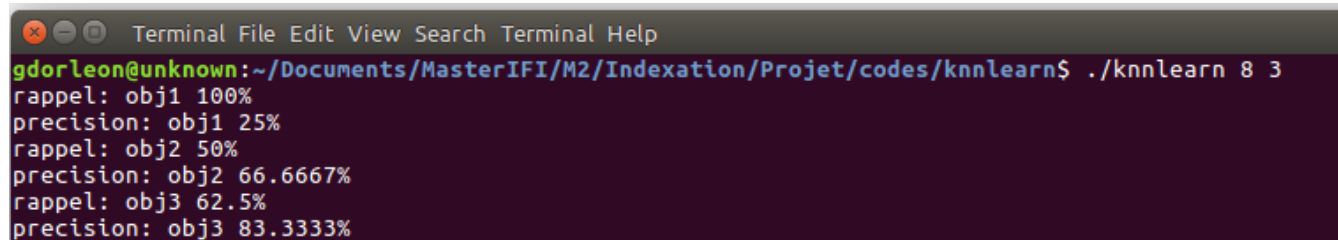
./knnlean 8 3

Notre programme d'évaluation utilise une base d'apprentissage et effectue les tests sur tous les éléments de la base test en utilisant l'algorithme knn. Puis une statistique est effectuée au fur et à mesure de l'évaluation des éléments de la base test pour calculer le rappel et la précision.

- En entrée, notre programme d'évaluation prend en entrée 2 paramètres:
 - 1- nombre de niveaux pour l'échantillonnage
 - 2- nombre de voisins
- En sortie, le programme renvoie comme résultat le calcul du rappel et de la précision du système pour toutes les classes d'objets.
-

II - Résultats de l'évaluation

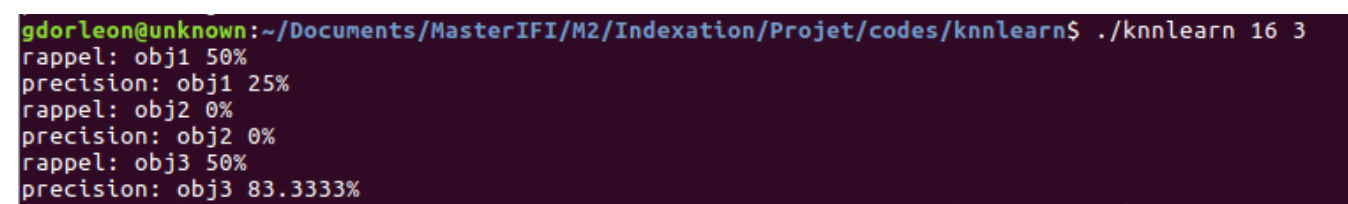
commande1 ./knnlearn 8 3



```
gdorleon@unknown:~/Documents/MasterIFI/M2/Indexation/Projet/codes/knnlearn$ ./knnlearn 8 3
rappel: obj1 100%
precision: obj1 25%
rappel: obj2 50%
precision: obj2 66.6667%
rappel: obj3 62.5%
precision: obj3 83.3333%
```

On constate qu'on obtient un pourcentage de 100% pour l'objet, ce qui traduit la bonne performance de notre système pour cet objet.

commande2 ./knnlearn 16 3



```
gdorleon@unknown:~/Documents/MasterIFI/M2/Indexation/Projet/codes/knnlearn$ ./knnlearn 16 3
rappel: obj1 50%
precision: obj1 25%
rappel: obj2 0%
precision: obj2 0%
rappel: obj3 50%
precision: obj3 83.3333%
```

On fait varier le nombre d'échantillon de 8 à 16, et on obtient encore un résultat intéressant pour l'objet 1. 100 % de rappel et de précision. Ce qui traduit une bonne performance pour l'objet 1 mais avec l'objet2, la précision est totalement nulle soit 0%. Ce qui valide le taux de 0% fourni par la matrice de confusion du bags-of-words démontré plus haut. Donc pour la classe 2 de la base de test, les deux évaluations fournissent le taux de 0%.

III – DISCUSSION, CONCLUSION GÉNÉRALE ET PERSPECTIVE

Si nous observons l'évolution des systèmes de recherche d'images par le contenu, nous remarquons que ces systèmes sont de plus en plus complexes. Dans ce travail, nous avons construit un système de recherche d'images par le contenu qui répond à de nombreuses difficultés telles que:

- Extraire des images les descripteurs visuels qui permettent de retrouver efficacement des images similaires,
- Trouver une représentation pertinente de ces descripteurs,
- Trouver une mesure de similarité efficace,
- Accéder rapidement à l'information
- Utiliser un système de sac de mots visuels.

Dans ce travail, on a pris connaissance de toutes ces méthodes pour une bonne application de ces systèmes en implémentant un système pour faire la recherche d'images dans une grande base de données. Des descripteurs globaux en allant jusqu'aux descripteurs locaux, notre système fait une utilisation efficace de tous ces paramètres.

En effet, notre système de recherche d'images par le contenu permet de rechercher les images d'une base de données en fonction de leurs caractéristiques visuelles. Dans ce système, la requête est une image et le résultat de la requête correspond à une liste d'images ordonnées en fonction de la similarité. Dans plusieurs domaines d'application, l'utilisation de descripteurs résumant l'information globale des images, tels que les histogrammes de couleurs des images entières, n'offre pas toujours des résultats satisfaisants car cette description ne tient pas compte de la localisation des pixels et des régions d'intérêt.

Comme perspectives à notre travail, nous tenterons d'extraire d'autres caractéristiques de l'image comme la texture, de les combiner afin d'obtenir une meilleure description de l'image améliorant ainsi la qualité de recherche future.

Références

- 1- Cours Indexation multimodale des contenus – Professeur **Mickaël Coustaty** – IFI -2017
- 2 - <http://www-ia.lip6.fr/~tollaris/ARTICLES/THESE/node6.html>
- 3 - .Huang J., Kumar S., Mitra M., Zhu W.J., Zabih R. (1997), “Image Indexing Using Color Correlograms”. Proc. of Conference on Computer Vision and Pattern Recognition (CVPR), San Juan (Puerto Rico), pp. 762-768, 1997.
- 4 - Maillot N., Thonnat M.(2005), “A Weakly Supervised Approach for Semantic Image Indexing and Retrieval”, International Conference on Image and Video Retrieval (CIVR), July 20-22, 2005, Singapore.

Notes aux lecteurs

1 - Toutes les codes sources de nos programmes peuvent être consultées gratuitement sur le lien suivant:

<https://github.com/gdorleon/Indexation#indexation>

2 – Les codes ne sont pas fournies sur papier dans ce rapport, pour les raisons suivantes:

Nous avons codé en C++ sur *gedit* et écrit un fichier de programme pour chaque option. Nous avons un total de 6 fichier.cpp d’environ 500 à 900 lignes de codes chacun, donc un seul fichier fait à lui seul 45 pages en pdf. Ce qui rend lourd le dossier qui empêche d’être mis sur le serveur moodle de l’IFI.

Donc, toutes les codes sources sont accessibles via le lien github donné plus haut.