Indexation des données multimédias

TP01 : Recherche d'images par contenu (CBIR)

Introduction

Dans ce tp, vous allez implémenter un petit système de recherche d'images par le contenu. L'idée est de calculer des caractéristiques pour chaque image et de rechercher les images ayant les caractéristiques les plus semblables. Le système que vous développerez se limitera à la recherche par comparaison d'histogrammes couleurs et par comparaison de textures (calculées par les matrices de co-occurrences). Il existe des solutions plus complètes, mais beaucoup plus complexes à réaliser pour le cadre d'un seul tp. Le système calculera la distance entre plusieurs images d'une base **(dossier base)** et requêtes **(dossier req)** et affichera la liste des images triées de la plus proche à la plus lointaine.

Aucune interface n'est prévue pour ce système. Tout se fera en ligne de commande et la sortie sera affichée sous forme texte vers la sortie standard. 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 (toutes les images contenues dans le répertoire courant). En sortie, le système affiche les N premières images (distance la plus courte) triées dans l'ordre. Un argument d'entrée permet de spécifier N (par défaut 10).

Le Langage de programmation sera le langage C/C++ avec la bibliothèque OpenCV (pour les outils de traitement d'images).

NB. La base d'images qui servira de test pour ce TP sera donnée lors de la séance de TP.

Buts: Extraire de l'information des images, comparer des images entre elles et évaluer le système.

I Recherche d'images par couleur(Descripteur : histogramme, distance=intersection d'histogrammes)

I.A Lecture des images de dossier base et de dossier req

4. **Écrire un programme qui lit les images de dossier base dans un vecteur de Mat et les images de dossier req dans un vecteur de Mat.**

I.B Calcul d'histogrammes des images des deux dossier

2. **Écrire une fonction qui calcule l'histogramme d'une image**

I.C Intersection (distance) de deux histogrammes

Pour chaque image, il y a un histogramme de taille 3 x 256 qui est calculé.

3. **L'étape suivante est de faire l'intersection entre histogrammes des images de la base (dossier base) et les images requête de test (dossier req) : float interHist(Histo_b, Histo_r)**

où Histo_r représente l'histogramme de l'image requête et Histo_b l'histogramme de l'image de la base. Le résultat de cette formule devrait être entre 0 et 1 et représente la distance couleur entre les deux images.

En appliquant la fonction sur toutes les images de la base et les images requête deux à deux, nous obtenons une **matrice de similarité** .

	image_b1	image_b2		image_bn
image_r1	d(imag_b1,imag_r1)			
image_r2				
image_rm				d(imag_bn,imag_rm)

4. **Dés matrice de similarité générée, pour tester votre application, dans une boucle infinie, on demande à l'utilisateur de saisir le numéro d'un fichier image(images requête)(dossier req). Ensuite, le système utilise la matrice de similarité pour affiche les N premières images similaire à l'image requête, triées dans l'ordre. Un argument d'entrée permet de spécifier le nombre d'image à retourner (N par exemple =3).**

II Recherche d'images par texture(Descripteur : uniformité, distance=distance de Manhattan)

II.A Calcul de matrices de co-occurences

La deuxième caractéristique utilisée pour la comparaison entre les images est la texture. Nous utiliserons les matrices de co-occurrences.

1. **Conversion de l'image en niveaux 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 (NdG) il suffit d'utiliser la fonction suivante (cvtColor).

2. **Écrire une fonction qui calcule matrice de co-occurrence d'une image (**distance=1 et pour des directions=45).

3. **Écrire une fonction qui calcule l'uniformité.**

II.B Calcul de la distance entre les textures de deux images différentes

5. **Écrire une fonction qui calcule la similarité entre images de la base et les images requête : dist(image_b, image_r)=| Param_b−Param_b|**

Param_b est l'uniformité calculée sur une image_b et **Param_r** est l'uniformité calculée sur une image_r.
En appliquant la fonction sur toutes les images de la base et les images requête deux à deux, nous obtenons une **matrice de similarité**.

1. **Testez votre programme**

III Fonction globale de similarité entre deux images

Nous avons maintenant deux distances : couleur et texture. La fonction globale de similarité entre deux images sera la somme pondérée de ces deux distances :

$$dist = poids * dist\_couleur + (1 - poids) * dist\_texture$$

où poids prend une valeur entre 0 et 1. Cette valeur peut être un nouvel argument de votre programme (valeur par défaut = 0.5).

pour tester votre application, dans une boucle infinie, on demande à l'utilisateur de saisir le numéro d'un fichier image(images requête)(dossier req). Ensuite, le système utilise la matrice de similarité globale pour affiche les N premières images similaire à l'image requête, triées dans l'ordre. Un argument d'entrée permet de spécifier le nombre d'image à retourner (N par exemple =3).