Projet de JAVA : Moteur de recherche d'images

Bertrand Le Saux

1 Présentation

Le but de ce projet est d'implémenter un logiciel de recherche d'images par le contenu. De la même manière qu'un moteur de recherche (altavista ou google par exemple) permet de rechercher un mot contenu dans des pages web sur l'internet, un logiciel de recherche d'images par le contenu permet de retrouver les images les plus ressemblantes à une image requête dans une base d'image.

Pour de plus amples renseignements sur le domaine, vous êtes invités à consulter le site web [1] de l'équipe Imedia de l'INRIA. Le logiciel Ikona en démonstration sur ce site vous donnera un exemple de moteur de recherche d'images.

Un tel logiciel peut avoir différentes applications, parmi lesquelles, dans le cas d'un systeme ouvert (le web par exemple), un moteur de recherche similaire aux moteurs traditionnels avec des mots, ou bien dans un système fermé : une agence de publicité disposant d'un catalogue de photographies cherche un paysage particulier pour illustrer une campagne publicitaire.

2 Description du logiciel

Un moteur de recherche d'images est composé de deux parties. L'ordinateur n'est pas capable de reconnaitre ce que representent les images. Il faut d'abord définir un moyen de décrire les images de manière automatique et permettant de les différencier. Ensuite, l'utilisateur doit avoir un moyen de chercher les images qu'il a en tête dans une base d'images. Le logiciel sera donc composé de deux outils correspondant à ces deux parties.

2.1 Indexation des images

Le premier outil à élaborer permet de calculer un index pour chaque image de la base. Un index est un vecteur d'entiers qui décrit une image, et qui peut être traite par l'ordinateur : pour comparer deux images, nous comparerons en fait leurs index associés. L'index que nous allons implémenter permet de décrire la proportion des couleurs de l'images.

2.1.1 L'espace de couleurs RGB

Une image numérique est une grille rectangulaire de pixels (points de l'image). Un pixel est un petit rectangle de taille fixe, doté d'une couleur. Cette couleur est codée en Java au

[1] http ://www-rocq.inria.fr/imedia/

moyen de quatre octets. Chaque octet peut être interprété comme une valeur entre 0 et 255. Le premier octet correspond à la transparence du pixel et ne nous intéressera pas dans la suite. Les trois derniers octets contiennent l'information de couleur : rouge pour le deuxième, vert pour le troisième et bleu pour le quatrième (RGB correspond aux initiales anglaises de ces couleurs). À partir de ces 3 couleurs de base, on peut recomposer toutes les couleurs de la palette.

2.1.2 Histogramme de couleurs

Un histogramme permet de mesurer la distribution statistique des couleurs : pour chaque couleur, on compte le nombre de pixels de cette couleur par rapport au nombre de pixels total :

$$h(\mathbf{c}) = \frac{1}{WH} \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} \delta\left(couleur(i,j), \mathbf{c}\right), \quad \forall \mathbf{c} \in C \tag{1}$$

Où δ représente le symbole de Kronecker : $\delta(x,y) = 1$ si $x = y$, et 0 sinon ; W désigne la largeur de l'image et H sa hauteur. Malgré l'apparente difficulté de la formule, le processus de calcul de l'histogramme est simple : on parcourt l'image pixel par pixel, et pour chaque pixel, on extrait son triplet RGB ($\in [0;255]^3$) puis on incrémente le compteur de pixels possédant ce même code RGB.

2.1.3 Quantification

En fait, on ne construit pas un histogramme pour les 255^3 couleurs possibles : en effet deux couleurs peuvent avoir des codes RGB sensiblement différents et pourtant être visuellement proches. On va donc quantifier l'espace de couleur afin de considérer des couleurs plus générales. Pour cela on divise chaque axe de l'espace de couleurs (rouge, vert et bleu) en un certain nombre N d'intervalles. la valeur de N fixe la précision de l'indexation : si N est trop petit, toutes les images auront des histogrammes semblables, si N est trop grand, les temps de calcul et l'espace occupé en mémoire deviennent trop importants. Une valeur $N = 4$ devrait être satisfaisante, ce qui donne $N^3 = 64$ couleurs dans l'histogramme.

2.2 Recherche d'images

Le deuxième outil doit permettre à l'utilisateur de faire sa recherche d'images, c'est-à-dire de choisir une image requête dans la base et de la comparer aux autres images pour y retrouver les images semblables.

2.2.1 Scénario de recherche d'images

Le logiciel présente des images choisies aléatoirement dans la base à l'utilisateur. Si l'utilisateur ne voit pas d'image qui correspond à ce qu'il cherche, un bouton permet de retirer de nouvelles images de manière aléatoire. Si l'utilisateur voit une image qui l'intéresse, il clique dessus, et le logiciel affiche les images classées par ordre décroissant de similarité. Si la base comporte des images semblables à la requête, elles sont donc présentées à l'utilisateur parmi les premières.

2.2.2 Interface graphique

L'interface graphique permettra d'afficher un aperçu des images de la base, c'est-à-dire $c*l$ images parmi celles de la base, c (respectivement l) représentant le nombre de colonnes (resp. de lignes) de l'affichage. Ces deux paramètres pourront être choisis par l'utilisateur. L'interface doit également permettre de cliquer sur une image pour retrouver les images similaires, et permettre le tirage aléatoire d'images dans la base. Enfin, l'utilisateur doit pouvoir régler les différents paramètres.

2.2.3 Calcul de similarité entre les images

Pour comparer les images, on va en fait comparer les index associés. Pour cela les histogrammes sont considérés comme des vecteurs dans un espace de grande dimension, et on peut alors calculer les distances entre l'histogramme de l'image requête et les histogrammes des autres images, afin de trouver les plus proches. On utilise la distance usuelle (la distance euclidienne), qui est définie par la formule :

$$d(u,v) = \sqrt{(\sum_{i=0}^{n-1}(u_i - v_i)^2)} \tag{2}$$

3 Travail demandé

3.1 Travail minimal

le projet final devra comporter au minimum :

- un premier outil d'indexation des images, qui :
  - chargera les images de la base,
  - calculera pour chaque image l'index (histogramme) associé,
  - écrira dans un fichier les différents index ;
- un deuxième outil de recherche d'image :
  - comportant une interface graphique telle que décrite en 2.2.2,
  - permettant de choisir l'image requête,
  - calculant les distances entre les images,
  - affichant les images de la base les plus similaires à la requête.

Après une première requête, l'utilisateur doit pouvoir recommencer une nouvelle recherche. La séparation du logiciel en deux outils présente l'avantage de ne pas refaire l'indexation (coûteuse en temps) à chaque nouvelle recherche d'images.

Pour écrire le programme complet, il est recommandé de créer de nombreuses classes, pour séparer les parties distinctes du projet. Par exemple, le calcul de distances doit se trouver dans une classe distincte de la partie graphique, qui peut elle-même être séparée en plusieurs classes.

3.2 Extensions

Plusieurs directions permettent de poursuivre et d'améliorer le projet (cela est vivement recommandé pour obtenir une bonne note) :

– D'autres distances sont plus adaptées que la distance euclidienne à la comparaison d'histogrammes de couleur. Tout d'abord la distance "Manhattan" définie par :

$$d(u,v) = \sum_{i=0}^{n-1} |u_i - v_i| \tag{3}$$

Puis la distance quadratique qui tient compte des similarités entre couleurs (rouge et rose sont plus proches que rouge et bleu). Il faut implémenter ces nouvelles distances et permettre à l'utilisateur de choisir la distance qu'il souhaite utiliser.

– pour trier la liste d'images, il est utile d'utiliser un algorithme de tri rapide. Voici la description de l'algorithme à implémenter pour classer une liste de nombres rapidement :

1. choisir un nombre dans la liste (le pivot) ;
2. séparer la liste en deux nouvelles listes, la première contenant tous les nombres inférieurs au pivot, la deuxième contenant les nombres supérieurs au pivot ;
3. recommencer ce qui vient d'être fait sur chaque sous liste ;
4. la liste triée est obtenue en concaténant les 2 sous-listes, en ayant mis le pivot au début de la deuxième.

Cet algorithme introduit la notion de récursivité.

– La couleur est le moyen le plus efficace de décrire une image. Cependant, dans certains cas, une information complémentaire sur les formes présentes dans l'image améliore les résultats. L'approche retenue ne recense pas des formes explicites (rond, carré..), mais tente de différencier les zones où la couleur est uniforme (et donc significative) et celles où la couleur ne correspond pas à une partie importante de l'image. On rajoute une pondération dans l'histogramme :

$$h(\mathbf{c}) = \frac{1}{WH} \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} w(i,j)\delta\left(couleur(i,j), \mathbf{c}\right), \quad \forall \mathbf{c} \in C \tag{4}$$

où $w(i,j)$ est par exemple le nombre de pixels du voisinage ayant la même couleur que le pixel courant.

– la possibilité d'avoir une requête externe : l'utilisateur fournit une image au système, qui calcule l'histogramme de cette nouvelle image et le compare aux histogrammes déjà calculés. Finalement les images similaires dans la base d'image lui sont retournées. Cette spécificité est nécessaire dans le cadre d'une utilisation comme moteur de recherche pour le web par exemple.

– selon votre imagination et les besoins qui vous apparaitront lors de la réalisation du projet...