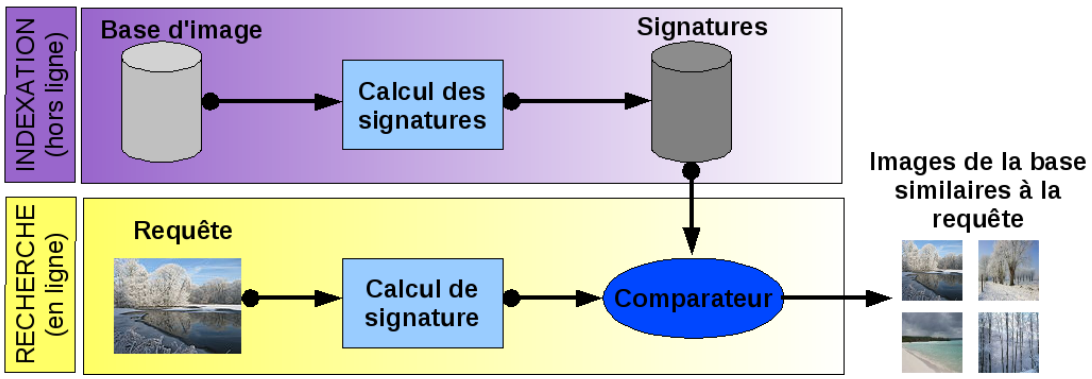

FIGURE 4 – Recherche d'images par le contenu (en anglais : Content Based Image Retrieval).

4 Sujet de projet : recherche d'images par le contenu

La recherche d'images par le contenu (en anglais : Content Based Image Retrieval ou CBIR) est une technique permettant de rechercher des images à partir de ses caractéristiques visuelles, c'est-à-dire induite de leurs pixels (par exemple l'histogramme de l'image). Un cas typique d'utilisation est la recherche par l'exemple où l'on souhaite retrouver des images visuellement similaires à un exemple donné en requête. Il s'oppose à la recherche d'images par mots clés ou tags, qui fut historiquement proposée par les moteurs de recherche tels que Google Image grâce à des banques d'images où les images sont retrouvées en utilisant le texte qui les accompagne plutôt que le contenu de l'image elle-même (Google Image propose désormais des filtres basés sur le contenu (pixels) des images).

Cette technologie s'est développée dans les années 90 pour la recherche de données dans les secteurs industriels, l'imagerie médicale ou cartographiques. Elle a donné lieu à de nombreux programmes et produits de la recherche. La reconnaissance faciale est utilisée par exemple par Interpol ou Europol pour rechercher les criminels. L'application la plus mature à la fin des années 2000 est la recherche de copie, utilisée dans la lutte contre la contrefaçon.

4.1 Principe

Le principe général de la recherche d'images par le contenu comporte deux étapes (voir la figure 4). Lors d'une première phase (**étape d'indexation**), on calcule les signatures des images et on les stocke dans une base de donnée. La seconde phase, dite de **étape de recherche** se déroule de la manière suivante. L'utilisateur soumet une image comme requête. Le système calcule la signature selon le même mode que lors de la première phase d'indexation. Ainsi, cette signature est comparée à l'ensemble des signatures préalablement stockées pour en ramener les images les plus semblables à la requête.

Lors de la phase d'indexation, le calcul de signature consiste en l'extraction de caractéristiques visuelles des images telles que : la texture, les formes, la couleur, etc. Dans le cadre de ce projet, nous utiliserons ici comme signature l'histogramme couleur d'une image.

Une fois ces caractéristiques extraites, la comparaison consiste généralement à définir une mesure de similarité entre deux images (nommée « comparateur » dans la figure 4). Au moyen de cette mesure de similarité et d'une image requête, on peut alors calculer l'ensemble des valeurs de similarité entre cette image requête et l'ensemble des images de la base d'images. Il est ensuite possible d'ordonner les images de la base suivant leur score, et présenter le résultat à l'utilisateur, les images de plus grand score étant considérées comme les plus similaires.

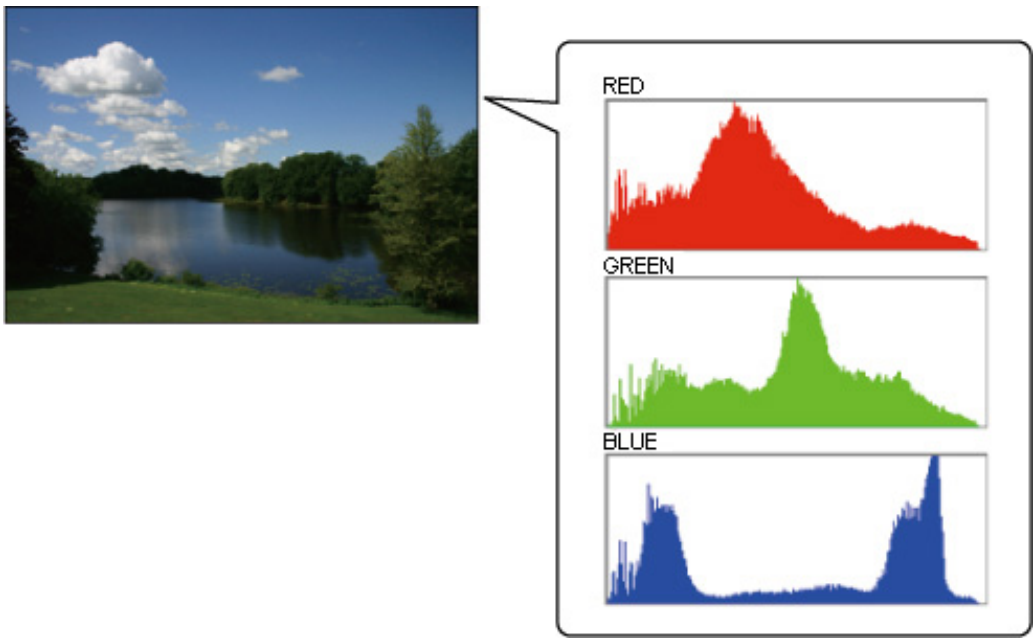

FIGURE 5 – Histogramme couleur d'une image.

4.2 Déroulement du projet

L'objectif du projet est d'implémenter un prototype permettant à un utilisateur d'effectuer une recherche d'images par le contenu. Cette implémentation Java sera réalisée à l'aide de la librairie PELICAN et des fonctions déjà codées durant les 3 premières séances de TP.

Partie 1 : premier prototype A partir d'une image requête fournie en entrée par l'utilisateur, le système devra rendre en sortie (sous la forme d'un affichage) les n images de la base de données les plus similaires à l'image requête, triées par ordre de similarité. Nous nous focaliserons ici uniquement sur les images en couleur. Afin de simplifier le problème, la signature d'une image sera ici modélisée par l'histogramme colorimétrique de l'image et nous n'implémenterons pas, dans un premier temps, le système d'index (les signatures seront calculées *à la volée* pour chaque nouvelle requête). Deux bases d'images que vous pouvez utiliser pour vos tests sont disponibles à l'adresse suivante :

`http://www.camille-kurtz.com/teaching/IUT/M4105C/Projet/images/`

Vous êtes libres également d'utiliser d'autres bases d'images ou de compléter ces dernières. Voici les étapes principales du prototype à réaliser :

1. **Lecture en mémoire d'une image requête** R. Appliquer un filtre médian sur cette image afin de potentiellement la dé-bruiter ; Afficher cette image avec le `Viewer2D` ;
2. **Construction de l'histogramme couleur** H_R de l'image requête : ⚠ on considère ici une image en couleur RGB (et non à niveaux de gris), il faut donc modifier la fonction de calcul d'histogramme implémentée précédemment en TP pour permettre la construction d'un histogramme couleur. Un histogramme couleur sera représenté par une matrice $H[256][3]$ contenant 3 lignes (une pour chaque canal R, G et B) et 256 colonnes (voir Figure 5). Un histogramme couleur est donc un histogramme à 3 dimensions.
3. **Discrétisation de l'histogramme** : Un histogramme classique est composée de 256 barres par canal (une barre par niveau de gris possible). Cette dimension est élevée et peut causer

des problèmes dans la fonction de comparaison lors de la recherche d'images similaires (non prise en compte de la corrélation entre des valeurs proches d'intensité). Pour pallier à ce problème, une stratégie consiste à réduire le nombre de barres de l'histogramme en discrétisant (diminuant) l'espace des valeurs. Coder une fonction, prenant en paramètre un histogramme $H[256][3]$, permettant de discrétiser un histogramme en divisant par 10 le nombre de barres de l'histogramme : la première barre codera les 25 premiers niveaux d'intensité, la deuxième barre codera les 25 niveaux suivants, etc.

4. **Normalisation de l'histogramme** : Pour que 2 histogrammes issues de 2 images de tailles différentes soient comparables, il est nécessaire de les normaliser par rapport au nombre de pixels dans l'image. Coder une fonction, prenant en paramètres un histogramme ainsi que le nombre de pixels dans une image, permettant de normaliser un histogramme en divisant chaque barre de l'histogramme par le nombre de pixels dans l'image afin d'obtenir une valeur de pourcentage.
5. **Recherche des images similaires** : parcourir à l'aide d'une boucle le dossier stockant la base d'images (en prenant soin d'ignorer l'image requête R si cette dernière est dans le dossier) :
   (a) Pour chaque image I de la base, appliquer un filtre médian sur cette image afin de potentiellement la dé-bruiter, puis construire l'histogramme couleur H_I de l'image (le discrétiser et le normaliser) ;
   (b) Calculer la similarité entre les 2 images R et I en mesurant la distance de similarité entre H_R et H_I (voir aide ci-dessous) ;
   (c) Stocker dans une structure de données maintenue triée (`TreeMap` par exemple ...) le nom de l'image I ainsi que sa valeur de similarité avec l'image requête R.
6. Afficher par odre de similarité les 10 premières images les plus similaires à l'image requête R (en égalisant au préalable leurs histogrammes). Attention la distance Euclidienne utilisée ici est une mesure de dissimilarité : les images de plus grand score sont considérées comme les plus dissimilaires.

Mesure de la similarité entre histogrammes
Dans de nombreuses applications, il est utile de calculer une distance (ou plus généralement une mesure de similarité) entre histogrammes. On définit pour cela une mesure pour calculer si deux histogrammes sont « proches » (se ressemblent) l'un de l'autre. Soit H_1 et H_2 deux histogrammes de même taille N (i.e. avec le même nombre de barres). La distance la plus utilisée est la distance Euclidienne :

$$d_{Eucl}(H_1, H_2) = \sqrt{\sum_{i=1}^{N} (H_1^i - H_2^i)^2}$$

où H_1^i représente la i-ème barre de l'histogramme H_1. C'est exactement le même principe que quand vous avez 2 points $p_1(x_1, y_1)$ et $p_2(x_2, y_2)$ dans le plan Euclidien $\mathbb{R}^2$ et que vous voulez connaître la distance entre ces 2 points via la formule $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Plus la valeur de la distance entre 2 histogrammes sera grande moins ces histogrammes seront similaires et inversement. La distance entre un histogramme et lui-même est nulle.
Pour mesurer la similarité entre 2 histogrammes couleur à 3 dimensions, on calcule indépendamment les distances euclidiennes entre les histogrammes représentant le canal R, G et B, puis on somme ces 3 valeurs de distance.

Partie 2 : utilisation d'un système d'indexation Afin d'éviter de recalculer à chaque fois (pour chaque nouvelle requête) l'ensemble des signatures (des histogrammes) des images du jeu de données, on pourra implémenter un système d'indexation basique. Pour ce faire, vous devrez rajouter au prototype déjà implémenté un pré-traitement qui parcourra chaque image de la base et qui

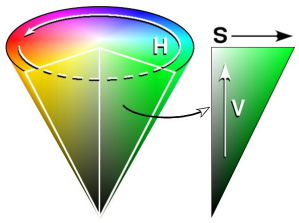

FIGURE 6 – Espace de couleur HSV (Hue-Saturation-Value).

calculera son histogramme couleur (normalisé et discrétisé). Ces histogrammes devront alors être stockés en mémoire physique (sous la forme d'un fichier texte ou dans une base de données SQ-Lite par exemple). Pendant l'étape de recherche d'images similaires, il ne sera donc plus nécessaire de calculer l'histogramme de chaque image de la base d'images mais simplement de récupérer en mémoire (ou en base de données) son histogramme, diminuant ainsi les temps de calculs lors de la recherche.

Partie 3 : utilisation d'un autre espace couleur La couleur est le descripteur visuel le plus employé, car c'est le plus perceptuel. L'espace RGB est très simple à utiliser, car c'est celui employé par de nombreux appareils de capture d'images qui effectuent leurs échanges d'informations uniquement en utilisant les triplets (R,G,B). Cependant, ces trois composantes sont fortement corrélées (par exemple, si l'on diminue la composante verte, la teinte parait plus rouge), de plus l'espace RGB est sensible aux changements d'illumination, et ne correspond pas au processus de perception humaine. L'espace HSV (Hue-Saturation-Value, voir la Figure 6) sépare les informations relatives à la teinte (Hue), la saturation (Saturation) et l'intensité (Value). Cet espace est plus intuitif à utiliser car il correspond à la façon dont nous percevons les couleurs. La teinte décrit la couleur (rouge, vert ...), la saturation décrit l'intensité de la couleur, et la valeur décrit la luminosité de la couleur.

Le travail demandé consiste ici à étendre le prototype pour permettre de retrouver des images similaires en se basant sur une signature d'image, toujours à base d'histogramme, mais dans l'espace HSV. Reprendre les étapes précédentes (Partie 1), où les images seront maintenant décrites par un histogramme couleur H_R calculé en considérant l'image dans l'espace HSV. Attention dans cet espace, les composantes V et S ne varient plus entre 0 et 255 mais entre 0.0 et 1.0. Par ailleurs, la composante H est exprimée en degrés et varie entre 0 et 360.

Pour la conversion d'un pixel de l'espace RGB à l'espace HSV, voir le lien ci-dessous :

`http://www.had2know.com/technology/hsv-rgb-conversion-formula-calculator.html`

Partie 4 : évaluation de la qualité du système Vous disposez maintenant de 2 systèmes différents permettant de retrouver (et d'ordonner), à partir d'une image requête, les images les plus semblables dans une base d'images. Pour être en mesure d'évaluer la qualité et les performances de votre système, il est nécessaire d'évaluer les résultats obtenus par ce dernier qui correspondent aux images retrouvées. Pour ce faire, la première étape consiste à regrouper l'ensemble des images de la base en différentes catégories, suivant le contenu de ces dernières (paysages de montagne, photos de plages, etc.). Ensuite, étant donnée une nouvelle image requête R (par exemple un paysage de montagne), et un ensemble d'images résultats retrouvées par le système, on peut évaluer la qualité de ces résultats en comptant le nombre d'images retrouvées (parmi les 10 premières) qui appartiennent à la catégorie *paysages de montagne*. Ce chiffre correspond à la précision du système pour la requête R et peut être exprimé sous la forme d'un pourcentage. En considérant

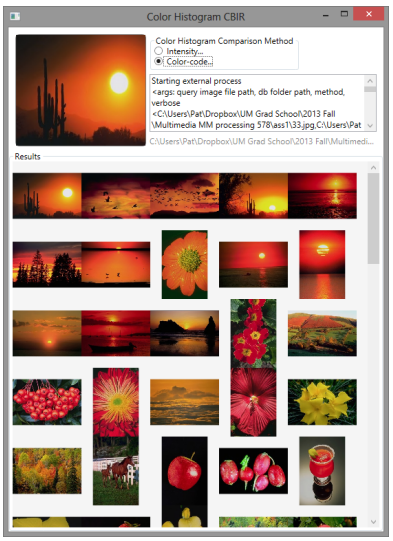

FIGURE 7 – Exemple d'une interface graphique permettant de visualiser l'image requête et les images similaires retrouvées dans la base.

chaque image de la catégorie *paysages de montagne* comme une nouvelle requête et en calculant la précision pour chacune d'elle, on peut alors obtenir la précision moyenne du système pour la catégorie *paysages de montagne*. D'autres mesures sont possibles

Le travail demandé consiste ici à :

- Créer un ensemble de catégories d'images (paysages de montagne, photos de plages, etc.) à partir de toutes les images de la base ;
- Développer un système permettant de calculer pour chaque catégorie un score de précision moyen ;
- Comparer les résultats obtenus avec votre prototype de CBIR dans l'espace RGB et celui dans l'espace HSV ; quelles catégories ont été les mieux reconnues ?
- Comparer les résultats obtenus avec votre meilleur prototype et celui des autres binômes (quel le meilleur gagne !) ;
- Evaluer vos résultats avec d'autres mesures comme la R-Précision, la Mean Precision et la Mean average precision (voir `https://en.wikipedia.org/wiki/Information_retrieval`).

Partie 5 : interface graphique Une fois que toutes les étapes précédentes ont été validées, vous pouvez réaliser une interface graphique de base (en Java ou en générant dynamiquement une page Web html) permettant de visualiser l'image requête et les images similaires retrouvées dans la base. La figure 7 présente un exemple d'interface graphique.

4.3 Consignes

- Projet à faire en binôme **(pas de trinôme)**.
- Évaluation et présentation en dernière semaine de TD. A l'issue de la présentation, le code source (.zip) devra être envoyé par email à votre enseignant de TP.
- Pour la présentation, chaque étudiant(e) devra apporter une version compilée du projet (qui fonctionne sur les machines de l'IUT) et réaliser une démonstration du programme devant un encadrant de TP.
- Toute absence injustifiée entraînera une note nulle au projet.
- **Afin de détecter une triche potentielle, les projets seront comparés via un outil automatique de détection de fraudes.**