Identification d'images pour la botanique

Présenté par : Issolah Mohamed

Encadrents :

Frédéric precioso

Diane Lingrand

Tuteur :

Olivier Corby

Jury :

Hélène Collaviza

Catherine Faron

Jean-Yves Tigli

Contents

List of Figures

1 Introduction

Nous traitons dans ce stage le problème de recherche d'image par contenu sur une base d'images issue du concours ImageClef. En premier lieu, je vais détailler notre participation au concours imageClef et la solution proposée en présentant les différentes techniques de la littérature. Je présenterai ensuite les différents aspects d'implémentation et l'API développée qui sera utilisée pour les prochaines participations. Je terminerai par une conclusion et les perspectives à venir.
Le stage est effectué au sein de l'équipe KEIA du laboratoire I3S qui se compose de 6 chercheurs; encadré par 2 encadreurs, Madame Diane Lingrand et Monsieur Frédéric Precioso.
Pour le bon déroulement un bureau m'a été attribué avec un ordinateur et un accès aux serveurs de calcul titan et bastion.
Dans ce rapport qui suit, je vais détailler les différents aspects de mon stage, je vais présenter les techniques et méthodes utilisées pour le problème du fossé sémantique, notre participation au concours ImageClef 2013, le travail effectué au sein de l'équipe et la valeur ajoutée.

2 Description du projet

Notre travail s'inscrit dans une logique plus générale qui consiste à développer une application mobile permettant à un botaniste de prendre plusieurs clichés de la même plante et les envoyer à un serveur pour vérifier si cette plante est déjà répertoriée.
Le projet peut être divisé en deux parties:

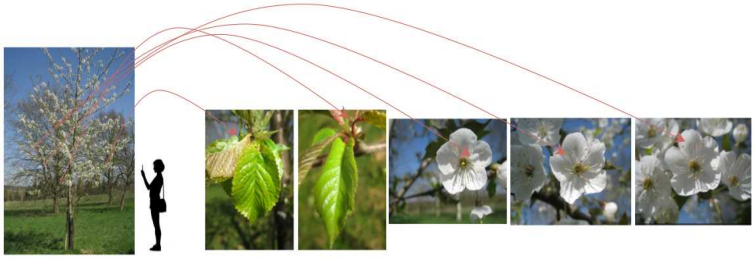


Figure 1: Vue générale du projet

- Développer l'application web avec l'interface graphique adéquate
- Développer le moteur d'identification de plante.

Notre travail se positionne sur la deuxième partie: la création d'un moteur d'identification d'image à partir d'un ensemble de photos de plantes.
Le domaine d'application du stage est la vision artificielle et plus précisément, la

recherche d'image par contenu (CBIR: Content Based Image Retrieval). Dans notre cas les images recherchées représentent exclusivement des plantes.
Une requête de recherche représente un ensemble de clichés d'une plante (voir figure 2).
Plusieurs points seront abordés pendant le stage.

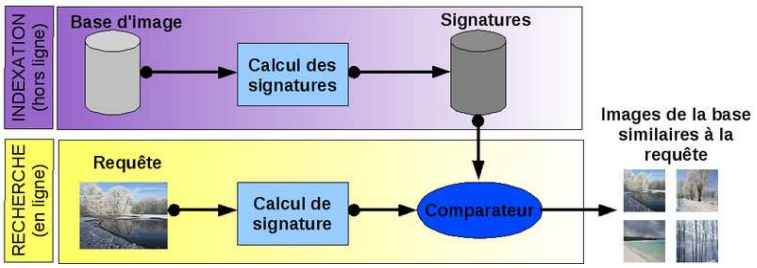


Figure 2: Recherche d'image par contenu

1. La complexité, non seulement le temps de calcul mais aussi l'occupation mémoire.

2. Le fossé sémantique entre la représentation d'une image sur une machine et son interprétation (descripteur d'image).

3. Machine Learning / Data Mining.

A la différence d'un moteur de recherche d'image traditionnel tel que *google search image*, qui pour une image de plante donnée, fournit comme résultat un ensemble de photo de plantes, notre système doit pouvoir trouver l'espèce de plante recherchée. Vue la complexité de l'identification de plante et l'intérêt de la communauté scientifique, un concours international ImageClef est organisé chaque année pour évaluer les différentes techniques proposées par les équipes de recherche. La participation à ce concours va être le fil conducteur du projet à moyen et à long terme.
L'autre conséquence de notre participation au concours ImageClef est le développement d'une solution modulable à long terme pour nos futures participations. Dans la partie suivante, je vais vous présenter les différents concepts de traitement d'image et l'état de l'existant.

3 État de l'art et outils utilisés

Dans ce qui suit, on prend en considération des images niveaux de gris, donc leur représentation sera en deux dimensions.

3.1 Point d'intérêt

Les points d'intérêts ou primitives de l'image sont les points qui caractérisent le mieux le contenu d'une image. On détecte ces derniers comme étant les points de contours de plus forte courbure. Dans les formes simples telles que le carré, les points d'intérêts sont facilement détectables à l'œil nu (voir figure 3). Et

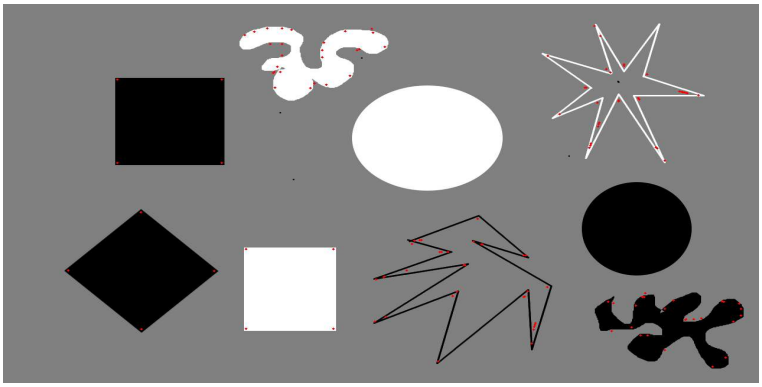


Figure 3: Points d'intérêt pour des formes simples

pour des formes plus complexes (voir figure 4) il existe plusieurs méthodes pour

Figure 4: Points d'intérêt sur une image de plante

la détection des points d'intérêt. Je vais présenter la méthode utilisée dans notre projet: *Harris corner detector* et une variante *Harris color detector*.

3.1.1 Harris Corner Detector

Cette méthode est utilisée sur les images en niveaux de gris.
L'algorithme est le suivant:

1. On calcule les dérivées premières à partir des dérivées de gaussienne (écart-type σ_D)
   (a) Lisser une image à l'aide d'une gaussiene.
   (b) Calculer la dérivée (ex: utiliser le filtre de Sobel)
2. On calcule les termes de la matrice d'auto-corrélation Ξ en calculant une moyenne locale des dérivées sous la forme d'une gaussienne (écart-type σ_I, typiquement $\sigma_I = 2\sigma_D$) $\Xi = \sigma_I * \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}$
3. Critère de Harris: $det(\Xi) - \alpha * Trace(\Xi)^2$ (typiquement $\alpha = 0,06$)
4. On prend les points supérieurs à un seuil.

L'étape 1.a consiste à appliquer un filtre gaussien pour lisser l'image et supprimer le bruit (voir figure 5). L'étape 1.b consiste à appliquerle filtre de Sobel

Figure 5: Application d'un filtre gaussien sur une image de plante

pour détecter les contours de l'image. Le filtre de Sobel s'applique par convolution de l'image avec deux noyaux 3x3 (voir figure 6).

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

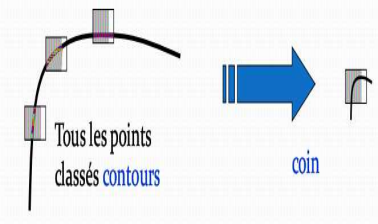
Figure 6: Convolution de l'image A avec les noyaux sur la direction X et Y

Après la convolution on aura les contours de l'image (voir figure 7).

Figure 7: Application d'un filtre de Sobel sur les directions X et Y

Les propriétés du détecteur de Harris sont les suivantes:

- Invariance aux translation, rotation, changement affine d'illumination
- Pas d'invariance à l'échelle.



Le détecteur de Harris est un des plus utilisés dans le domaine de la vision artificielle. Il existe une variante pour les images couleurs: *Harris Color Detector*.

3.1.2 Harris Color Detector

Pour une image I:

- On calcule la matrice d'auto-corrélation pour l'image des composantes rouges M_r
- On calcule la matrice d'auto-corrélation pour l'image des composantes bleues M_b
- On calcule la matrice d'auto-corrélation pour l'image des composantes vertes M_g
- On somme les 3 matrices $\Xi = \begin{pmatrix} R_x^2 + G_x^2 + B_x^2 & R_xR_y + G_xG_y + B_xB_y \\ R_xR_y + G_xG_y + B_xB_y & R_y^2 + G_y^2 + B_y^2 \end{pmatrix}$
- On applique le critère de Harris sur la matrice résultante. Cette méthode est une variante de Harris Corner Detector mais sur chaque composante de couleur de l'image.
  Le Harris Color Detector est une nouvelle méthode de détection de point d'intérêt et elle n'est pas beaucoup utilisée dans la littérature scientifique.

Pour résoudre le problème de la variance par rapport à l'échelle, je vais présenter la notion d' *Espace d'échelle*.

3.2 Espace d'échelle

L'Espace d'échelle (Scale space) représente les différentes échelles de détails qu'on peut voir dans une image. L'espace d'échelle se trouve dans un intervalle:

- inner scale: taille du plus petit objet perceptible dans l'image.
- outer scale: taille maximale d'un objet dans l'image.

Figure 8: Espace d'échelle d'une photo de forêt

Dans la plupart des applications de vision, on ne connaît pas à l'avance l'espace des échelles de notre environnement. C'est pour cela que dans la phase de traitement d'image, on doit prendre en considération la formalisation de l'espace des échelles en utilisant la notion de multi-échelle (voir figure 9). Les

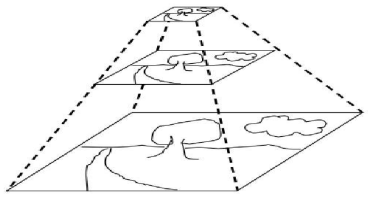


Figure 9: Multi-échelle

échelles sont obtenues grâce au filtre gaussien, en changeant le paramètre σ. Les différence de gaussiennes vont nous permettre de faire ressortir les détails qui existent entre différentes échelles.
On peut voir le résultat d'une différence de gaussienne dans la figure 10.

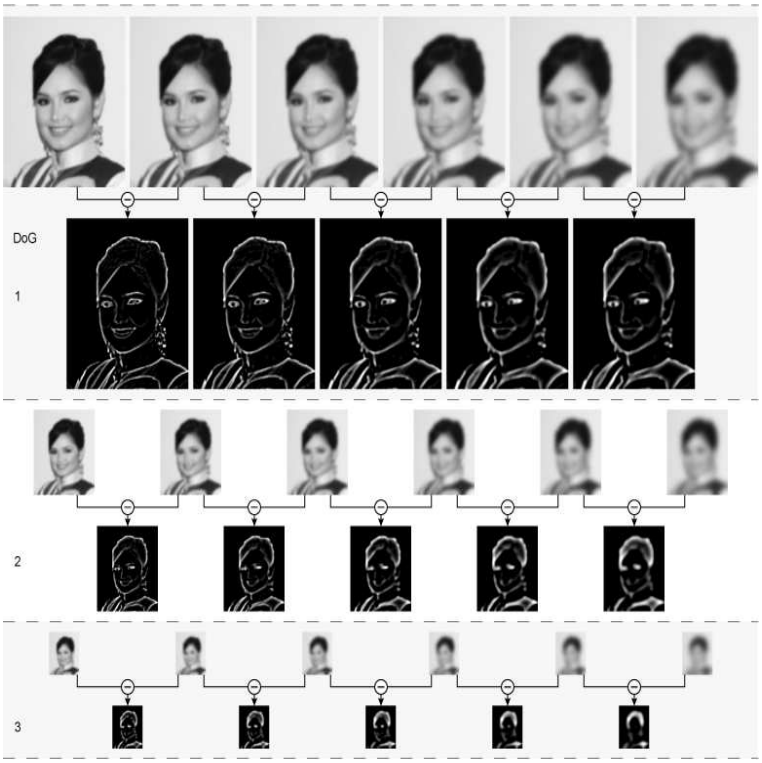


Figure 10: Différences de gaussiennes

3.3 Descripteur visuel local d'image

La représentation matricielle est purement informatique et ne permet pas de dégager de la sémantique. Pour cela, les descripteurs visuels locaux sont utilisés pour permettre une meilleure représentation de la sémantique de l'image (couleur, texture, ...).
Un descripteur visuel local est la description du voisinage d'un point d'intérêt. Les descripteurs d'images sont une partie cruciale pour les algorithmes d'identification, ils seront utilisés dans les différentes étapes de l'identification.

3.3.1 SIFT

Un descripteur SIFT est un vecteur d'entier de taille 128 qui décrit une zone 16x16 autour d'un point d'intérêt. Un descripteur SIFT n'est pas sensible à l'orientation, ni au changement d'échelle. Les différentes étapes de la génération des descripteurs SIFT sont

1. Un point d'intérêt (x, y, σ) est défini d'une part par ses coordonnées x et y,

et d'autre part par son facteur d'échelle qui est déterminé par la présence du point dans l'espace d'échelle, aux échelles $\sigma/2$, σ, et2σ.

2. On divise l'espace autour de chaque point d'intérêt (x, y) en 4x4x4. (voir figure 11 image 2)
3. On calcule le gradient $(G_x(a, b, \sigma), G_y(a, b, \sigma))$ pour les 4x4x4 points (a,b).
4. Pour chaque carré 4x4, on calcul un histogramme des orientations quantifiées en 8 directions, en pondérant par:
   (a) le module du gradient
   (b) l'inverse de la distance au point d'intérêt (x,y).
5. Pour être invariant à la rotation: l'orientation locale du point d'intérêt $\theta(x, y)$ est utilisée comme origine d'orientation des gradients.

La figure 11 présente les différentes étapes de la méthode SIFT.

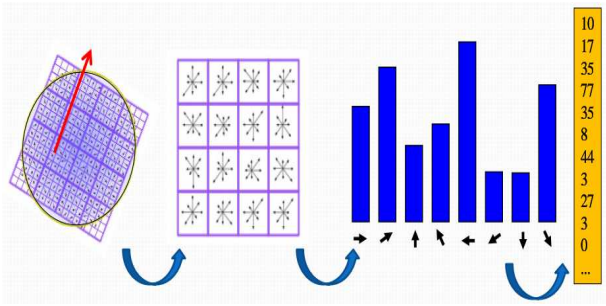


Figure 11: Etapes de la méthode SIFT

Pour obtenir ce vecteur, on considère autour d'un point d'intérêt, une région de 16x16 pixels, subdivisée en 4x4 zones de 4x4 pixels chacune. Sur chaque zone est calculé un histogramme des orientations des gradients d'intensité (orientation des contours dans l'image), voir figure 12.
L'histogramme comporte 8 intervalles (somme des gradients à 0, 45, 90, 135, 180, 225, 270, 315 degré), ce qui génère un vecteur de 16 blocs associés à des vecteurs de taille 8.
Un descripteur SIFT n'est pas sensible à l'orientation de l'image, ni au changement d'échelle (zoom). En d'autres termes, pour la même image tournée ou zoomée, les descripteurs seront les mêmes.

3.4 Apprentissage non supervisé

L'apprentissage non supervisé ou clustering est une méthode qui permet de classer les données dans des clusters (classes) différents en respectant les deux critères suivants:

Figure 12: Les points d'intérêt avec leurs orientations et amplitudes

- Minimiser la distance entre deux éléments de la même classe (distance intra-classe)
- Maximiser la distance entre les classes (distance inter-classe)

Dans un algorithme d'apprentissage le but est de minimiser le taux d'erreur avec un temps d'exécution raisonnable.
Je vais présenter l'algorithme d'apprentissage non supervisé utilisé pour notre projet.

3.4.1 K-Means

Méthode itérative qui divise l'ensemble des données en K classes. Elle se base sur le calcul de la distance entre les éléments et la mise à jour des barycentres des K clusters. C'est à l'utilisateur de fixer le nombre K de classes (clusters). L'algorithme K-Means est le suivant:

1. choisir k objet(s) formant ainsi k clusters.
2. (Ré)affecter chaque objet O au cluster C_i de centre M_i tel que $dist(O, Mi)$ est minimale
3. Recalculer M_i de chaque cluster (le barycentre)
4. Aller à l'étape 2 si on vient de faire une affectation

La figure 13 représente un exemple d'exécution de K-Means avec 2 classes.

3.5 Apprentissage supervisé

A la différence de l'apprentissage non supervisé, dans le cas de l'apprentissage supervisé l'ensemble des données est divisé en 2 parties:

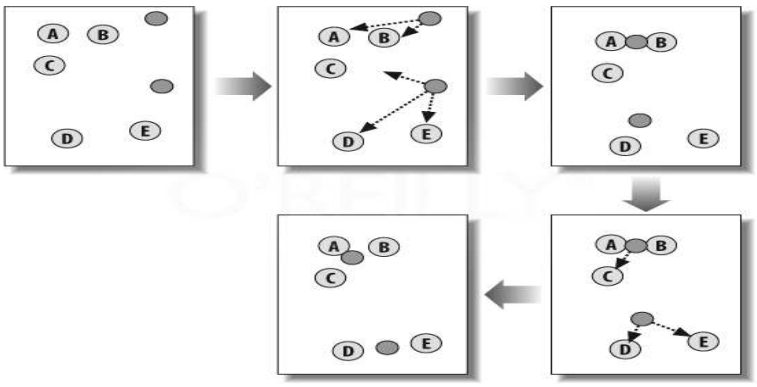


Figure 13: Exemple d'éxecution de l'algorithme K-Means avec 2 clusters

- Données d'apprentissage: un apprentissage a priori est effectué sur ces données avec une intervention de l'humain.
- Données de test: utilisées pour vérifier la validité de notre apprentissage.

L'apprentissage supervisé est utilisé pour la reconnaissance des formes et la prédiction.
L'algorithme utilisé dans notre projet est la Machine à vecteurs de support.

3.5.1 Machine à vecteurs de support (SVM)

SVM (Support Vector Machines) est un séparateur linéaire. Il est utilisé dans le cas des problèmes à 2 classes.
Dans l'exemple de la figure 14, initialement, les données ne sont pas classées. Après l'exécution du SVM, l'ensemble des données est divisé en deux classes: les points et les carrés. L'efficacité d'un SVM réside dans la maximisation de la marge (la variable w), à la différence d'un réseau de neurones où le but est de trouver la ligne séparatrice entre les deux ensembles (ici la ligne noire), un SVM maximise la marge qui est la distance entre la ligne verte et la ligne rouge sans pour autant augmenter le taux d'erreur.

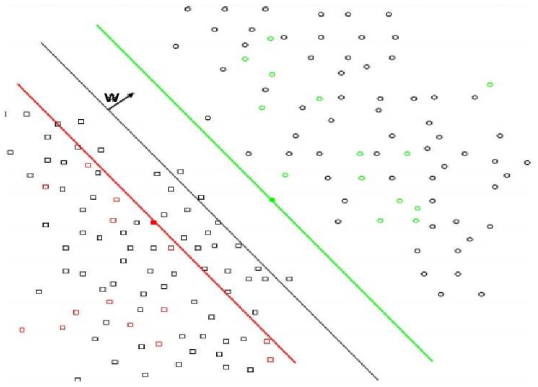


Figure 14: Séparateur linéaire entre deux ensembles: les carrés et les points

3.6 RANSAC

Cet algorithme permet de trouver à partir d'un nombre de points d'intérêt de deux images, s'il existe une correspondance entre un ensemble de points d'intérêt de la première image et un ensemble de points d'intérêts de la deuxième image (voir figure 15). Les differentes étapes de l'algorithme RANSAC:

1. Sélectionner aléatoirement un sous-ensemble des données d'origines
2. Construire un modèle à base de ces points
3. Cherche les autres points qui appartiennent à ce modèle et les ajouter à l'ensemble s'il s'ajuste avec une erreur inférieure à t
4. si le nombre d'éléments de ce modèle est supérieur à d (paramètre de l'utilisateur), le considérer comme un modèle candidat.
5. Après k itérations, prendre le meilleur modèle des modèles candidats.

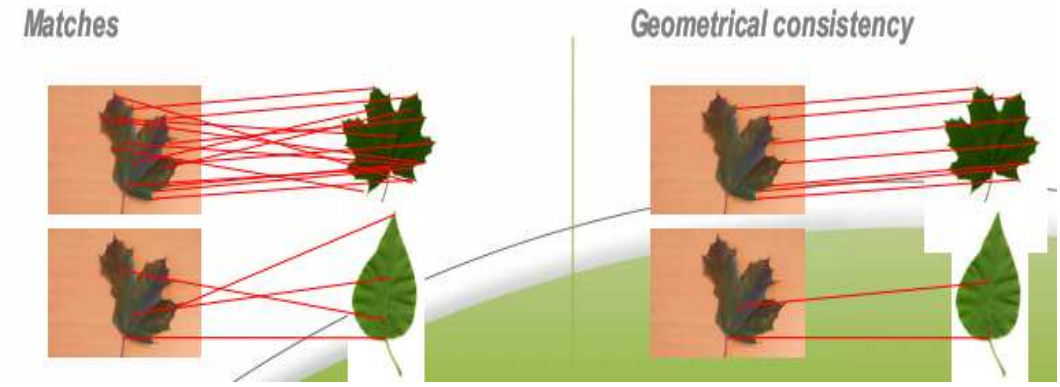


Figure 15: Trouver la consistance géométrique entre la forme des feuilles (RANSAC)

3.7 LSH

Avant d'introduire LSH, on parlera du problème des K plus proches voisins[1] (KNN), qui consiste à trouver dans un espace les K plus proches voisins d'un point i. La notion de "proche" est relative à la notion de distance entre deux points qui peut être définie selon l'espace.
Dans notre cas, on a un espace euclidien, la distance euclidienne entre deux points est utilisée. Dans ce cas la solution la plus simple est de calculer la distance euclidienne de tous les points de l'espace avec le point i et sélectionner les K plus proches voisins. Voir figure 16 Il existe plusieurs méthodes pour réduire

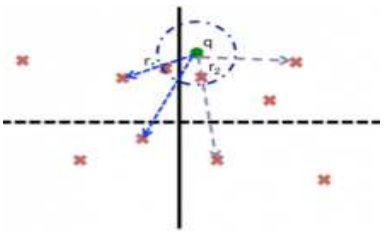


Figure 16: Les deux plus proches voisins du point q

la complexité des calculs, l'une d'elle est LSH. La technique Locality Sensitive Hashing (LSH)[AGM99] est utilisée pour diminuer la quantité d'informations à traiter dans le cas du problème KNN. Elle est basée sur un découpage de l'espace dans lequel se trouvent les points et sur l'idée que deux points "proches" dans l'espace avant le découpage doivent se retrouver dans la même cellule après

[1] https://fr.wikipedia.org/wiki/Méthode_ des_ k_ plus_ proches_ voisins

découpage (on dit aussi entrer en collision) alors que deux points éloignés doivent être dans des cellules différentes. D'un point de vue plus formel:

- Dans une espace métrique $M = (E, d)$ (où E est un ensemble non vide, d distance définie sur les éléments de l'ensemble E)
- On définit la famille LSH F avec un seuil R et un facteur d'approximation $c > 1$.
- F est une famille de fonctions $h : M- > S$ telle que, pour 2 points $p, q \in M$ et une fonction de hashage h choisie aléatoirement:

  1. si $d(p, q) =< R$ alors $P_{r_{h \in F}}[h(p) = h(q)] >= P_1$. En d'autres termes, si la distance entre 2 points est inférieures à un seuil R alors la probabilité pour qu'ils se trouvent dans la même case doit être supérieur à P_1

  2. si $d(p, q) >= cR$ alors $P_{r_{h \in F}}[h(p) \neq h(q)] <= P_2$

- Si $P_1 > P_2$, la famille F est appelée (R, cR, P_1, P_2)-sensitive
- Plus $P_1 > P_2$ plus la famille est intéressante.

D'une manière plus claire, le but est de faire une transformation de la présentation des données pour diminuer la complexité de calcul. Dans la figure 17, on divise notre espace avec plusieurs plans choisis aléatoirement mais d'une manière homogène. Pour chaque plan on regarde si le point se trouve au dessus ou au

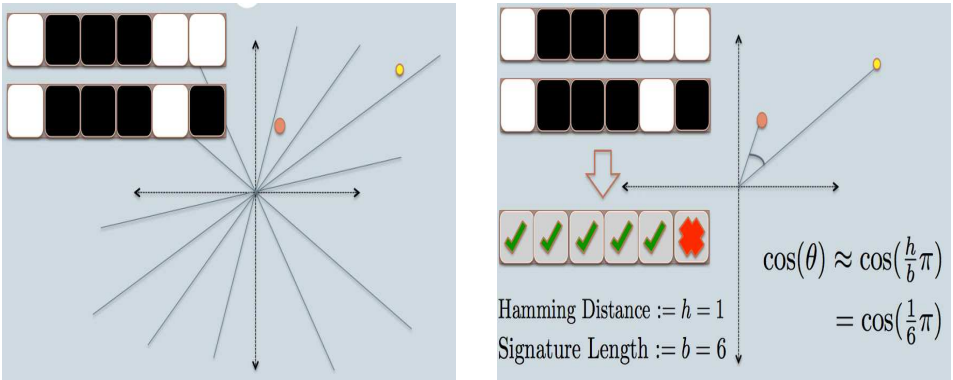


Figure 17: Local sensitive hash

dessous du plan. A la fin de toutes les opérations, chaque point est représenté par un vecteur binaire d'une taille équivalente au nombre de plans. La distance entre 2 points est la distance de Hamming qui représente la nombre de cases où la valeur est différente.

3.8 *Bog-of-words*(Bow)

Le *bag-of-words* est une méthode de génération d'index de recherche pour les moteurs de recherches d'image. Je présente l'algorithme général de cette méthode qu'on détaillera par la suite.

1. Extraire les points d'intérêt de chaque image.
2. Décrire chaque point d'intérêt avec un ou plusieurs descripteurs: à ce moment pour chaque image, on a un ensemble de descripteurs pour décrire ces POI.
3. Mettre tous les mêmes types de descripteur dans un même ensemble, et exécuter l'algorithme de clustring K-means pour classer chaque descripteur dans son cluster adéquat.
4. Chaque cluster est représenté par un centroïde.

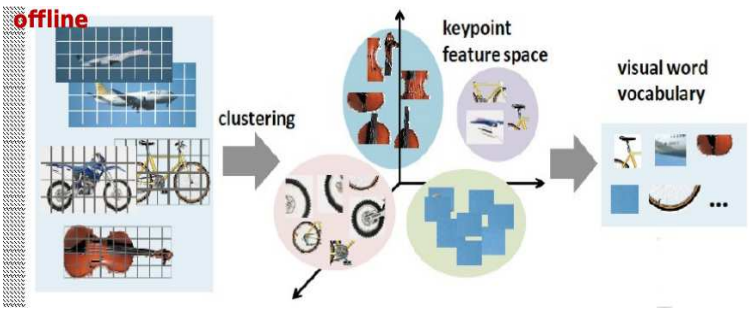


Figure 18: Génération du vocabulaire

5. Le descripteur d'une image est un histogramme où les variables sont les centroïdes des clusters et la répartition est le nombre de descripteurs de l'image dans le cluster représenté par le centroïde.

C'est un modèle issu de la recherche textuelle, qui consiste à générer l'ensemble des mots visuels caractérisant chaque image en se basant sur un dictionnaire, vocabulaire ou visuel.
Le vocabulaire est l'ensemble des barycentres de nos classes générés après le clustering des descripteurs de toutes les images. Le Bag-Of-Words (BOW) d'une image est un vecteur dont la J^{ieme} composante contient le nombre de descripteurs visuels locaux de l'image qui ont été classés (par le clustering) dans le cluster J. Dans l'exemple de la figure 19, l'histogramme du BOW sera transformé en un vecteur des fréquences des barycentres. La génération de l'index est faite en mode offline. Pour une requête d'image, les étapes 1, 2 et 5 sont effectuée pour générer son histograme.

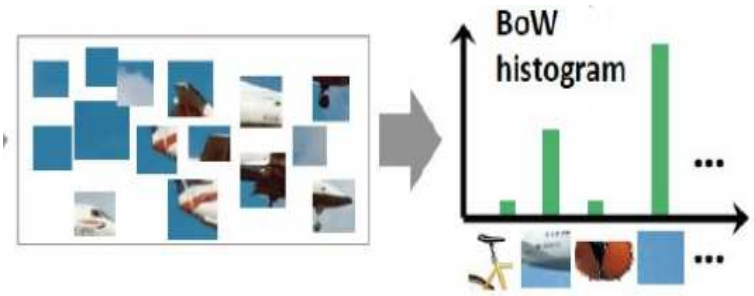


Figure 19: Génération des histogrammes de chaque image

4 Contribution

Dans cette partie, je vais vous présenter le travail réalisé au sein du stage, notre contribution à ImagineClef 2013 et l'API développée.

4.1 ImageClef 2013

ImageCLEF[2], est un concours d'identification de plantes organisé chaque année depuis 2003. Il nous donne l'opportunité de tester nos méthodes sur une grande base de données (26077 images) déjà annotée ainsi que la comparaison avec les algorithmes concurrents.
L'énoncé du concours est simple: pour une image de plante requête, trouver sa classe correspondante (la classe est le nom de l'espèce). Les détails de la base sont les suivants:

- 26077 images
- 20985 images d'apprentissage associées à un fichier XML qui décrit leur classe (classID)
- 5092 images de test sans classID
- 250 classes de plantes
- Au maximum 5 organes pour la même plante:
  - Feuille (16% du total de la base)
  - Fleur (18% du total de la base)
  - Fruit (8% du total de la base)
  - Tronc (8% du total de la base)
  - Arbre (8% du total de la base)

[2]http://www.imageclef.org/

En plus des organes, les photos sont divisées en 2 catégories selon leur arrière-plan:

- SheetAsBackground: exclusivement pour les feuilles, un fond blanc.
- NaturalAsBackground: fond naturel.

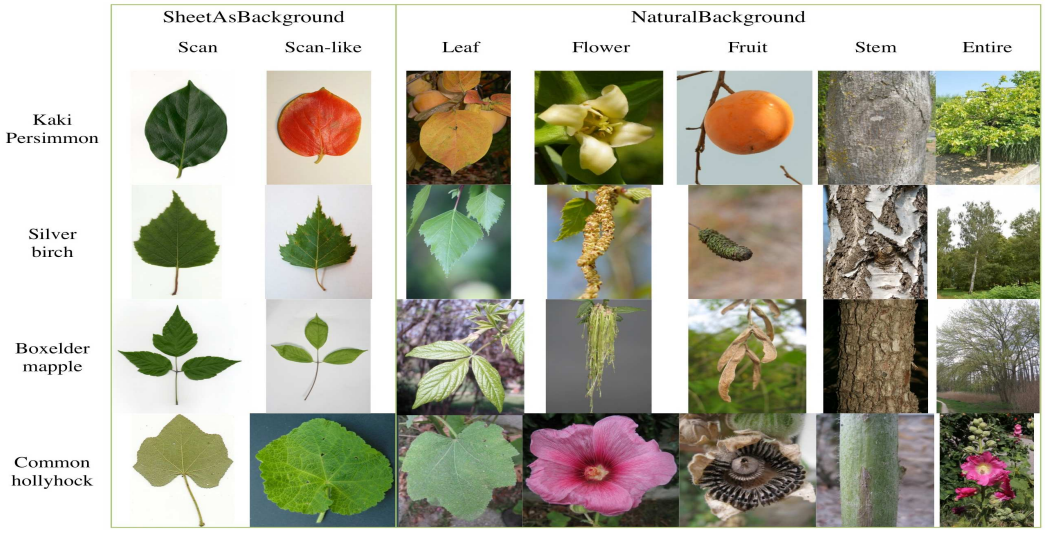


Figure 20: Exemple: 2 catégories avec 7 sous catégories

Chaque image est associée à un fichier XML contenant différentes informations telles que:

- classID: donnée présente seulement dans les images d'apprentissage, elle représente le nom de la plante
- Type: SheetAsBackground ou NaturalAsBackground
- Content: L'organe de la plante.

4.1.1 Chaîne de résolution

Lors de la première partie de mon stage, j'ai implémenté une chaîne de traitement complète afin de traiter les images du concours ImageCLEF. Cette dernière peut être utilisé pour le traitement des données du Laos[3] au sein du laboratoire. Je vais maintenant détailler la chaîne de traitement.
Le système implémenté repose sur le "bag-of-words" avec cette configuration

- Les points d'intérêts sont détectés et décrits avec SIFT.

[3]http://www.biotik.org/species_ list_ laos.html

- Le clustering pour la génération de l'index est fait avec l'algorithme K-means de 100 clusters.

On utilise les méta-données des fichiers XML pour extraire les informations comme: le type de contenu, l'arrière plan de l'image.
Le système est divisé en 3 modules:

1. Extraction et description des images
2. Apprentissage
3. Test

La complexité de l'application causée par les différentes étapes et la quantité d'information à gérer (pour une image donnée, le nombre de points d'intérêt est d'environ 1000) est très importante. Le but a été de minimiser le temps d'exécution de l'application en diminuant les itérations au maximum, et d'un autre côté éviter la saturation de la mémoire vive due à la taille des données.
En outre, l'application a été fragmentée pour permettre l'exécution de chaque étape sur une machine différente. Le flux d'informations à transmettre d'un module à un autre est stocké dans des fichiers (par exemple les descripteurs des images ou les SVMs).
Pour la première étape, un vocabulaire est créé pour chaque organe. Après cette dernière, on a l'ensemble des descripteurs pour chaque image avec le vocabulaire de chaque organe. La prochaine étape est l'apprentissage automatique sur ces donnée selon deux critères: la classe et l'organe.
La méthode d'apprentissage utilisée est la machine à vecteur de support (SVM). La même configuration des SVM(s) est utilisée pour les différents organes avec le paramètre $C = 20$, qui représente la somme de la distance des erreurs.
A la fin de la seconde étape, nous obtenons un SVM selon la classe de la plante et sa catégorie:

- Entire
- Stem
- Fruit
- Flower
- Leaf NaturalBackground
- Leaf SheetAsBackground

Les SVM(s) sont organisés en vecteurs selon leurs catégories (entire, stem, fruit ...) comme sur la figure 21. La taille d'un vecteur est de 250 qui correspond au nombre de classe dans nos données. La dernière étape est le test, qui consiste à rechercher pour une image donnée sa classe.
Les étapes à suivre sont:

1. Extraire la catégorie à partir du fichier XML.

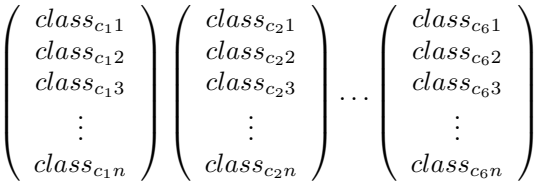


Figure 21: Chaque vecteur correspond à une catégorie (n = 250)

2. Générer le descripteur en utilisant le vocabulaire de la catégorie déjà créé——.
3. Exécuter les SVM(s) du vecteur correspondant à la catégorie
4. Le résultat est la classe du vecteur avec le meilleur score.

La figure 22 représente notre système dans sa globalité.

4.1.2 Implémentation

Le choix de la librairie OpenCV[4] (Open Computer Vision: librairie de traitement d'images développée en C++) était évident de par la maturité de la librairie, les différentes fonctions proposées et sa stabilité. Elle propose différents descripteurs tel que SIFT et SURF et différentes méthodes d'apprentissage tel que SVM.
Par contre, pour le langage, nous avons opté au départ pour le langage Python permettant un développement et une prise en main rapide d'OpenCV.
Cependant, toutes les fonctionnalités d'OpenCV n'étant pas disponibles depuis Python, une alternative se présente:

- écrire un wrapper Python pour les fonctionnalités manquantes (ex: BOW)
- passer en C++

Nous avons opté pour le passage en C++ notamment pour des raisons d'efficacité. Différents outils Linux ont été utilisés comme GDB pour le debuggage, CMake pour la compilation et d'autres pour l'extraction des données à traiter et le formatage des résultats (script shell, awk, grep ...)
L'équipe KEIA dispose d'un serveur de calcul sous CentOS disposant d'une capacité de stockage de 25Go et de 23 processeurs . Cependant, j'ai été confronté à des problèmes de compatibilité entre les versions des librairies utilisées rendant le développement plus complexe (par exemple entre OpenCV et LibXML).

4.1.3 Resultat et analyse

Le but était de connaître les résultats obtenus en utilisant le descripteur SIFT qui est considéré comme l'un des descripteurs optimum.
Les résultats sont présentés sur les figures 23, 24 et 26.

[4] opencv.org

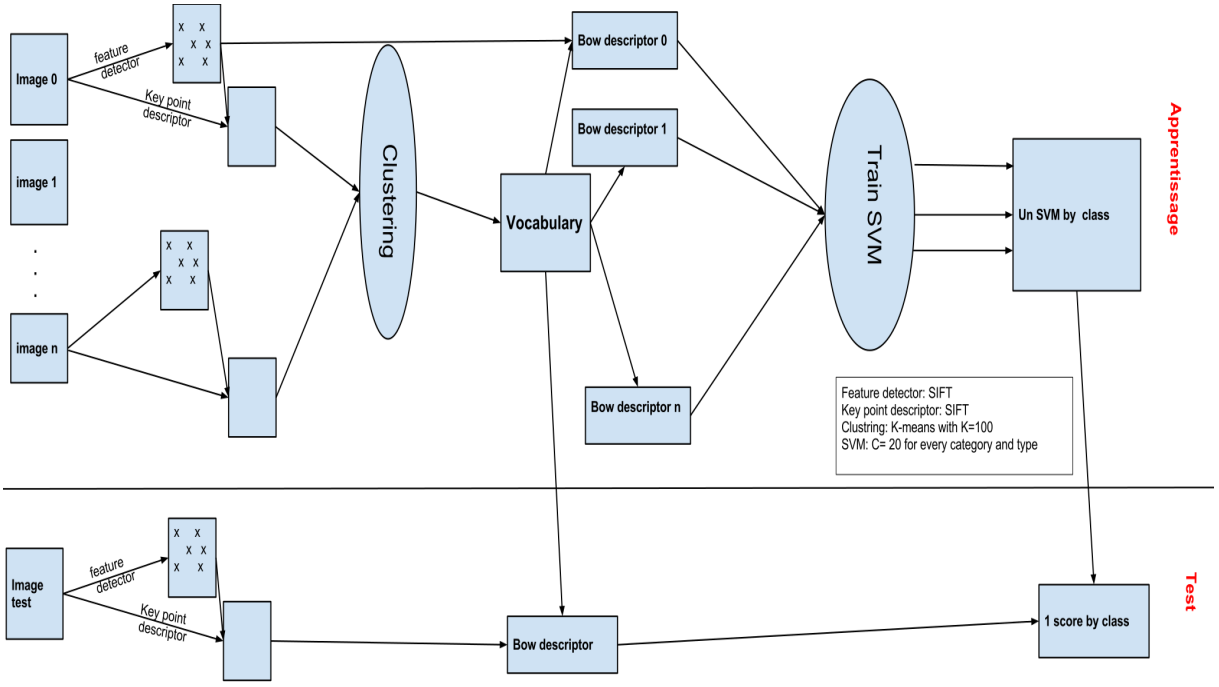


Figure 22: Schéma global de notre système

Run name	runfilename	Entire	Flower	Fruit	Leaf	Stem	NaturalBackground
I3S Run 1	1368034466828_new_100	0.017	0.023	0.041	0.038	0.025	0.026
I3S Run 2	1368165605197_new2_100	0.017	0.023	0.041	0.038	0.025	0.026

Figure 23: Resultat ImageClef 2013 selon la catégorie

Les résultat obtenus lors du concours ne sont pas satisfaisantes ce qui etait prévisible et ce pour deux raisons principales:

- La configuration des différents composant de notre système n'est pas optimisée.
- L'utilisation d'un seul descripteur SIFT.

Le manque de temps pour l'étude des différents paramètres (1 mois), nous a restreint à utiliser une configuration moins gourmande en espace mémoire et plus rapide à l'exécution. on pourrait utiliser d'autre descripteur tels que la texture et la couleur.
Cependant, cette participation nous a permis d'avoir les premières briques pour

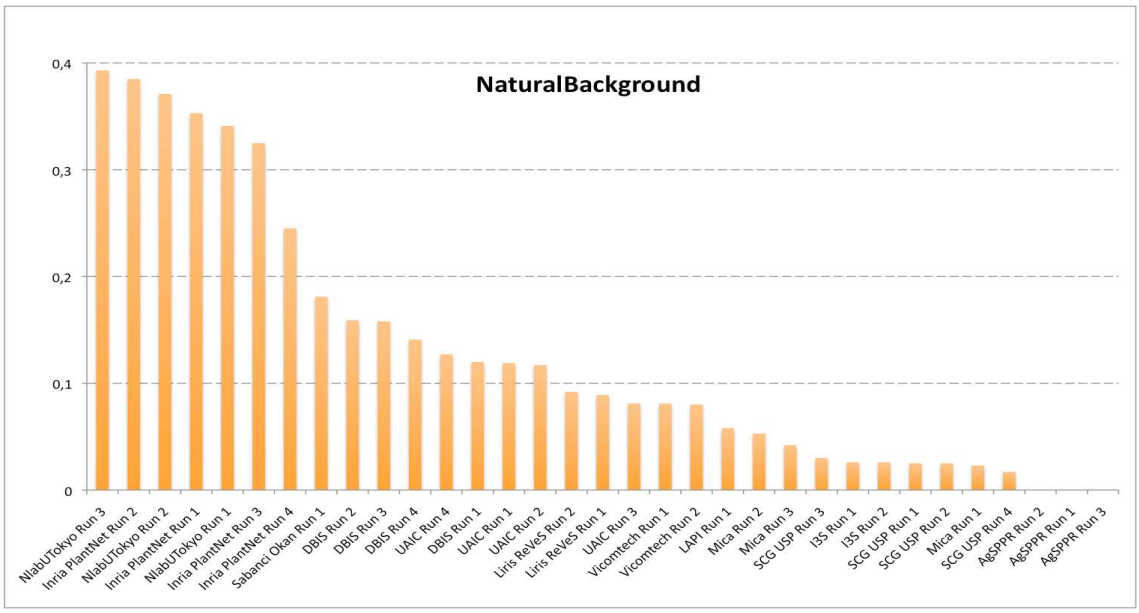


Figure 24: NaturalBackgroundScores

les prochaines participations.
De cette analyse découles le deux contributions suivantes à savoir:

- Développer une API pour faciliter l'implémentation et la modification des solutions pour les prochaines participations.
- Optimiser les paramètres de notre chaîne de résolution.

4.2 Développement de l'API

Dans la chaîne de résolution présentée précédemment, les différentes parties qui doivent être modulables sont:

- Les détecteurs des points d'intérêts.
- Les descripteurs.
- La méthode de clustring pour la génération du vocabulaire.
- La méthode d'apprentissage.

La figure26 présente le diagramme UML avec les différentes classes de notre application et l'API implémentée. Dans ce diagramme, les interfaces sont développées dans la libraire OpenCV

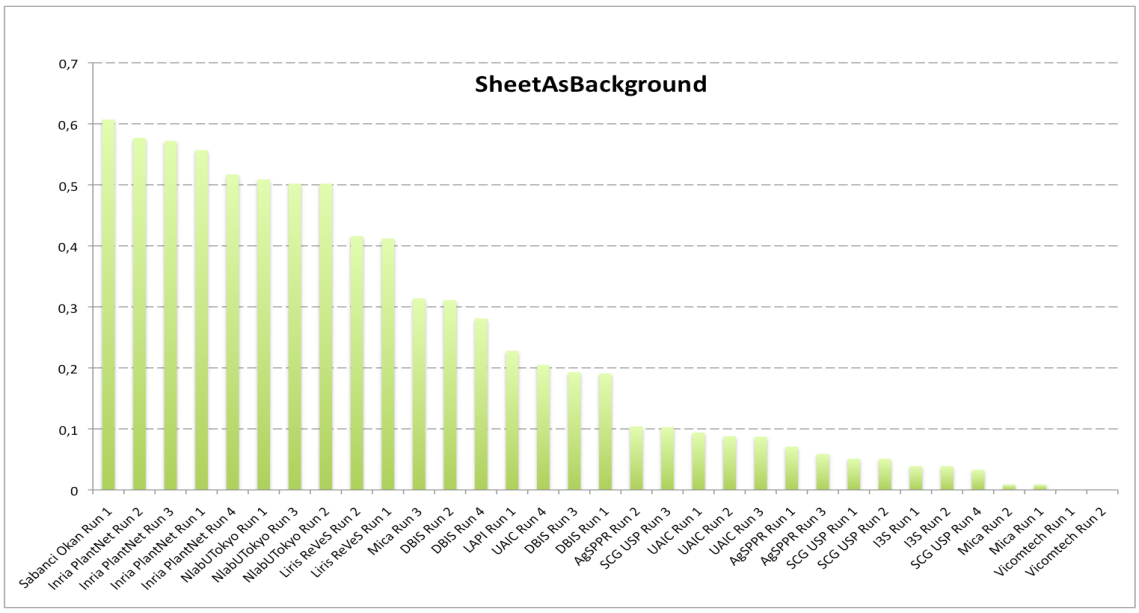


Figure 25: SheetAsBackgroundScores

- BOWTrainer: Chaque algorithme d'apprentissage qui génére le vocabulaire doit implémenter cette classe. Exemple: BOWKmeansTrainer pour la méthode K-Means
- CvStatModel: Chaque algorithme qui génère notre modèle tel que SVM doit implémenter cette classe. Exemple: CvSVM pour la méthode SVM
- featureDetector: implémenté par les détecteurs des points d'intérêt.
- DescriptorExtractor: implémenté par le descripteur de point d'intérêt.

Deux interfaces créées pour pouvoir implémenter notre chaîne de résolution:

- Data: Représente la donnée de notre moteur de recherche, dans notre cas les images de plantes. Elle doit implémenter les fonctions getPoi(), getDescriptor(), getBow() qui permettent respectivement de récupérer les points d'intérêt, leurs descripteurs et la génération du bag-of-words
- System: Représente notre système qui va générer le vocabulaire et le modèle grâce au fonctions: VOCGen(), training(). La fonction loadData() permet de charger les données dans la mémoire grâce aussi à ImageClefParser qui permet de parser les fichiers XML

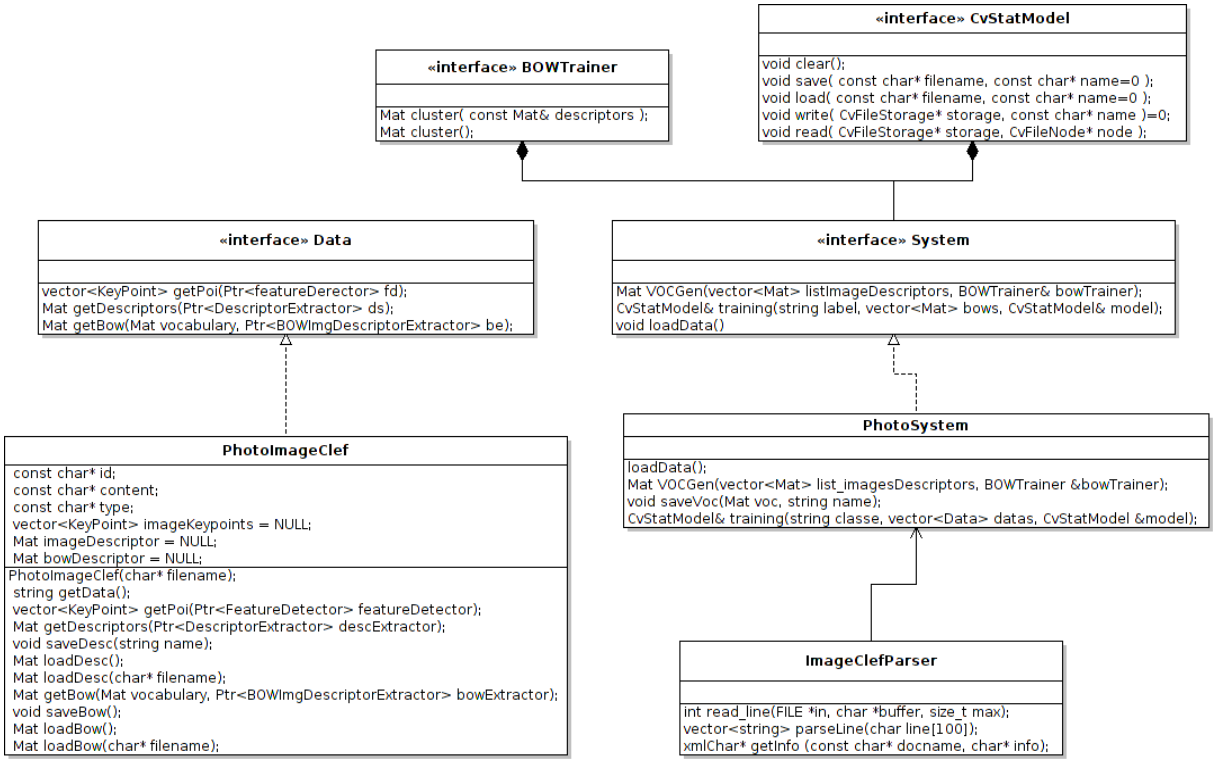


Figure 26: Diagramme

4.3 Optimisation des paramètres

Actuellement, nous attendons les résultats de cette phase. Le temps de calcul des différentes étape étant très importants. Je vais expliquer en quoi consistent les différents paramètres à traiter avec les valeurs de tests.
Ce qui est intéressant de noter est qu'à la différence de la solution proposée pour le concours où les paramètres sont les mêmes pour tous les organes. Le but cette fois est de trouver le paramètre adéquat selon l'organe. Un autre point abordé lors de cette partie est la répartition des données en données de test et d'apprentissage que je vais détailler par la suite. Les paramètres a tester sont:

4.3.1 Nombre de cluster

Le nombre de cluster du Kmeans: plus le nombre de cluster est importants, plus la taille de descripteurs augmente. La taille du descripteur influence sur le caractère **discriminant** de notre modèle.
En effet, il est intéressant de voir que la discriminalité est à prendre différemment selon l'organe.
Dans le cas des feuilles, on peut remarque sur la figure 27, que les trois feuilles

sont assez proches en terme de couleur mais reste assez différentes en terme de forme, où les deux premières appartiennent à la même classe. Les points d'intérêts des deux images vont être concentrés sur les contours de la feuille et aux centres. Par contre, dans la figure 28, on remarque que dans l'image il y a

Figure 27: organe feuille de la même plante

beaucoup de détails, on aura des points d'intérêt un peu partout dans l'image, et pas seulement sur l'arbre. Le caractère **discriminant** dans le cas des feuilles doit être supérieur à celle des arbres et ce à cause du nombre de points d'intérêt des feuilles inférieur à celui des arbres donc plus porteur d'informations que ceux des arbres.
Les valeurs étudiées pour les tests sont les suivantes: $100, 200, 500, 1000, 2000, 4000$;

Figure 28: photo complète (categorie entire) de la même plante

4.3.2 La somme de distance des erreurs du SVM (C)

Le paramètre C représente le taux d'erreurs accepté pour notre modèle. Plus le nombre est important, plus notre modèle est efficace. L'efficacité de notre modèle va à l'opposé du temps d'exécution.
Les valeurs étudiées pour les tests sont les suivantes: $0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100$

4.3.3 Les données

Un autre point important avant l'apprentissage est la problématique de la répartition des données entre données de test et données d'apprentissage. L'un des points importants qu'il faut prendre en considération est que la distribution des données d'apprentissage doit suivre le même modèle de distribution que les données de test. L'algorithme suivi pour la répartition des données se base sur l'information *IndividualPlantId* qui un identifiant stockée dans le fichier XML. Des fichiers XML qui ont le même *IndividualPlantId* représentent la même plante photographiée à plusieurs reprises.
Les étapes de l'algorithme sont comme suit:

- Regrouper les fichiers XML avec le même identifiant
- si le nombre de fichier XML pour le même identifiant est supérieur à 1, alors répartir la moitié dans le dossier de test et le reste dans le dossier apprentissage.
- sinon les mettre dans le dossier d'apprentissage

Avec cette répartition chaque espèce des plantes de test apparaitra dans les données d'apprentissages.

5 Conclusion

Etant étudiant en *Knowledge Information System*, ce stage m'a permis de voir les différentes techniques de *data mining* appliquée à des problèmes de vision artificielle et plus précisément la recherche par contenu. Les différents papiers que j'ai pu lire m'ont permis de voir la complexité d'un moteur de recherche d'images par contenu.
Le problème du fossé sémantique qui reste toujours posé et la configuration des différents paramètres de la chaine de résolution comme le nombre de clusters, le taux d'erreur, le filtre gaussien pour la détection de point d'intérets, le modèle à utiliser peuvent grandement influencer la qualité des résultats.
Comme perspectives, je citerai:

- Se rapprocher d'une équipe botaniste, voir comment ils reconnaissent et classent les plantes.
- A partir des connaissances botaniques créer différents descripteurs qui peuvent exprimer ces connaissances.
- Implémenter le descripteur de texture pour les troncs et des descripteurs de forme pour les feuilles.

6 Annexe

6.1 Présentation du laboratoire I3S

Le laboratoire I3S est une Unité Mixte de Recherche (UMR) de près de 300 personnes commune à l'Université de Nice-Sophia Antipolis (UNS), au Centre National de la Recherche Scientifique (CNRS) et à l'institut National de la recherche en Informatique et en Automatique. Il est composé en majorité d'enseignants-chercheurs de l'UNS (83 permanents et 17 associés) dans les sections 27 et 61 du Conseil National des Universités (CNU). Ceux-ci interviennent principalement dans les départements Informatique et Electronique de l'Ecole Polytechnique Universitaire (Polytech'Nice - Sophia) et de l'UFR Sciences, ainsi que dans les départements Informatique, Réseaux & Télécommunications, Génie Electrique et Informatique Industrielle de l'IUT. Ils portent dans ces structures d'enseignement de nombreuses responsabilités de filières, de spécialités, d'années de formation ou de départements.

6.2 Présentation équipe KEIA

Le groupe de recherche KEIA (Knowledge Extraction, Integration & Algorithms) est membre du Pôle GLC (Génie du Logiciel et de la Connaissance) du Laboratoire I3S. Les travaux entrepris dans ce groupe concernent la fouille de données, ou data mining, et ses applications. Ils ont pour objet l'extraction d'informations sémantiques à partir des données brutes par le développement

de techniques et d'algorithmes pour l'analyse de grands ensembles de données hétérogènes peu structurées ou non-structurées.

Bibliographie

References

[AGM99] Piotr Indyky Aristides Gionis and Rajeev Motwaniz. Similarity search in high dimensions via hashing. *Very Large Data Base Endowment*, 1999.

[ALP] P.H. Gosselin A. Lechervy and F. Precioso. Kernel combination by boosting for multi-classes classification of flower images. In *EEE International Conference on Image Processing*.

[BFT12] Elisa Fromont Basura Fernando and Tinne Tuytelaars. Effective use of frequent itemset mining for image classification. *European Conference on Computer Vision*, 2012.

[HGM] Julien Barbe Vera Bakic Alexis Joly Hervé Goëau, Pierre Bonnet and Jean-François Molino. Multi-organ plant identification. In *ACM international workshop on Multimedia analysis for ecological data*.

[HGM12] Alexis Joly Itheri Yahiaoui Daniel Barthelemy Nozha Boujemaa Hervé Goeau, Pierre Bonnet and Jean-François Molino. The imageclef 2012 plant identication task. *ImageCLEF*, 2012.

[PMD98] V. Gouet P. Montesinos and R. Deriche. Differential invariants for color images. *Institute of Electrical and Electronics Engineers*, 1998.

Résumé

La recherche d'image par contenu (CBIR) est un problème complexe dû au fossé sémantique entre la représentation et le contenu ou l'interprétation de l'image. La détection et description des points d'intérêts sont des méthodes pour réduire ce fossé. Notre domaine d'application est l'identification d'image de plante.
ImageClef est un concours à l'échelle internationale pour l'identification d'image de plante qui fournit une base de donné annoté d'une taille de: 20985. Cette base sera utilisée pour tester nos différentes solutions.
La première étape sera l'implémentation du bag-of-words avec les K-means comme méthode d'apprentissage pour générer les vocabulaires et SVM pour la création de notre modèle.
La seconde étape est de concevoir et développer un api qui permet d'implémenter n'importe quelle chaine de résolution se basant sur le bag-of-words.
La dernière étape est de voir le comportement de notre modèle en modifiant different paramètre pour trouver la configuration optimum.

Abstract

The image search by content (CBIR) is a complex problem due to the semantic gap between the representation and the content or interpretation of the image. The detection and description of interest points are methods to reduce this gap. Our scope is to identify image plant. ImageCLEF is a competition at the international level for the image identification of plant that provides a basis for a given annotated size: 20985. This database will be used to test our various solutions.
The first step will be the implementation of bag-of-words with the K-means as a method of learning to generate vocabularies and SVM to create our model. The second step is to design and develop an API that allows you to implement any string of resolution based on the bag-of-words. The last step is to see the behavior of our model by modifying different parameters to find the optimum configuration.